ULTRIX

Kernel Messages Reference Manual

Order Number: AA-PBKUA-TE
June 1990

Product Version: ULTRIX Version 4.0 or higher

digital equipment corporation
maynard, massachusetts

Contents

About This Manual

1 Kernel Messages

2 Kernel Panic Messages

About This Manual

This manual documents messages produced by the files in the ULTRIX kernel.

Audience

This manual is written for programmers experienced in using ULTRIX. DIGITAL field service and software support personnel will also find this manual useful when responding to customer's system problems. The messages documented here serve as a starting point for resolving hardware-detected and software-detected problems that are reported through the ULTRIX kernel software.

Users of this manual are expected to have access to the ULTRIX source code for this version of the operating system.

Organization

This manual contains two chapters and one appendix.

- Chapter 1, Introduction

  Defines the kernel, introduces the kernel messages, and describes the format of the panic messages.

- Chapter 2, Kernel Panic Messages

  Describes the panics in the ULTRIX kernel, in alphabetical order.

- Appendix A, Kernel Files

  Lists the files in the ULTRIX kernel in a directory tree format.

Related Documents

The *Guide to the Error Logger System* manual describes the setup and administration of an ULTRIX system. You should study the manual's discussion of the error logging facility, particularly in regard to:

- Using `uerf` ,the error report formatter, to extract reports about errors from the kernel errorlog file
- Maintaining the error logging facility with the `eli` command
- Configuring error logging for a system with the `elscd.conf` file
- Administering error logging and reporting between local and remote systems

Conventions

The following conventions are used in this manual:

`special`	In text, this type indicates the exact name of a command, variable, option, partition, pathname, directory, or file. This type is also used to indicate output, to the console subsystem log or to the user, that is associated with some messages.
`<0Xd>`	A hexadecimal number in output associated with the kernel messages.
`<0d>`	An octal number in output associated with the kernel messages.
`<d>`	A decimal number in output associated with the kernel messages.
`<"string">`	An ASCII string in output associated with the kernel messages.

Kernel Messages 1

This chapter introduces the kernel messages by defining the ULTRIX kernel, describing how messages are classified, produced, and reported, and showing the format of error messages.

1.1 The ULTRIX Kernel

Routines in the ULTRIX kernel produce kernel messages. In this manual, the kernel is defined as the set of files shipped as the base ULTRIX operating system for the Version 4.0 software release. Appendix A shows these files outlined in a directory tree format.

This manual does not include messages from optional products, even though some optional products, when installed on the ULTRIX base system, add files to the kernel. For example, when the optional product DECnet/ULTRIX is installed on an ULTRIX base system, the kernel is rebuilt and some DECnet/ULTRIX files become part of the new kernel. These DECnet/ULTRIX files, and the messages they may produce, are not included here.

1.2 Kernel Messages

Kernel messages result from software-detected situations that report problems with and provide information about the operating environment of the system. The following sections discuss how ULTRIX classifies, produces, and reports messages.

1.2.1 How Kernel Messages Are Classified

Messages are grouped into three levels based on their severity. The messages associated with the severity levels are called error messages (also called panics), warning messages, and informational messages. All conditions causing the messages are evaluated and reported by the kernel software. All errors are logged by the kernel software to the kernel errorlog buffer.

This manual does not include warning messages, or informational messages. Only error messages (see Section 1.3) are listed in Chapter 2.

1.2.1.1 Errors – An error results when a software-detected problem causes the software in the kernel to:

- Initiate (or not initiate) hardware recovery procedures
- Bring down (crash) the operating system
- Flag the hardware to rebootstrap the operating system

Data may be lost when a software-detected error occurs.

The ULTRIX *System Management Guide* contains information about procedures to follow for system crash recovery.

1.2.1.2 Warnings – Warnings show some situation or potential problem that the individual user, field service person, or system administrator should examine and perhaps resolve to keep a system operating with integrity.

Warnings may prevent the operating system from continuing operations, although they typically result in an error only when the problem they represent is not resolved.

1.2.1.3 Information – Informational messages are for all user levels. ULTRIX seldom produce informational messages from kernel files, although drivers may note operator information such as whether a device is off line or write-protected. Generally, informational messages are produced by utilities and shells.

1.2.2 How Kernel Messages Are Produced

When a kernel file detects an error or other event that requires a message, it calls the appropriate routine in the kernel file `/usr/src/sys/sys/subr_prf.c`. The routines in this file format the message and send it to the console subsystem or to the user. The routines do other things too, from basic housekeeping to rebooting the system.

In ULTRIX, ''hard-errors'' (for example, corruption or machine-irrecoverable errors) are called panics because the messages produced by the kernel files are effected through a call to the `panic.c` routine in either the `/usr/src/sys/machine/vax/panic.c` or the `/usr/src/sys/machine/mips/panic.c` kernel file. The routine does the following:

- Sends the message to the console (or console subsystem, depending on the processor), errorlog buffer, or both
- Calls the appropriate panic device and displays it
- Halts the appropriate processors
- Saves the state of the machine
- Flags the hardware to reboot the system, and determines the setup to reboot
- Saves the core and dumps it to swap space

1.2.3 How Kernel Messages Are Reported

Kernel messages are always reported to the console subsystem. When the ULTRIX error logging facility is active, the messages are also reported to the ULTRIX errorlog buffer. What happens to messages after they are reported to the errorlog buffer depends on how the error logging facility is defined and administered.

Depending on how reporting is enabled, the logical console, the console subsystem to which errors are reported, can be on the local system or at a remote system in a network.

Depending on how the error logging facility is defined, the reporting of errors can be extracted locally or remotely from the local system's kernel errorlog buffer.

1.3 Panic Messages Format

Chapter 2 contains descriptions of all the panics in the ULTRIX kernel, presented alphabetically. All panics have the same format: ''panic:'' followed by a brief message. The format for representing the message and its related information is:

message string

File	Name
Routine	Name
Problem	A brief description of the cause of the message
Output	Additional information associated with the panic message

The message string reproduces the message generated by the call to the `panic.c` routine, minus the ''panic:'' that precedes it. The file name gives the name of the directory and source file containing the routine that detected the problem. The routine name is the function in the source file that detected the problem and then issued the call to the `panic.c` routine. The problem section describes the situation that caused the panic. The output section gives the meaning of associated messages sent to the console log and some or all of the console output. The following example shows typical output in each category:

getegnode: free gnode isnt

File	/sys/gfs/gfs_bio.c
Routine	getegnode
Problem	A gnode on the free list is still active. This routine gets a gnode from the free list. When it does so, the routine checks the gnode's reference count, which is zero when a gnode is not active. In this case, the routine detected the reference count was not zero, indicating the gnode was still active.
Output	Indicates the routine and the gnode address and number. The format is: `getegnode: gp <0Xd> (<d>)`

1.3.1 Resolving Panic Message Problems

This manual does not provide specific information concerning the procedures necessary for you to resolve a particular panic. If you are unable to solve a problem that caused a panic, consult an ULTRIX Software Support Group. If this is not a viable solution, submit a software problem report (SPR), including a listing of the console terminal output, and machine readable copies of the following files:

- The system configuration file
- The system error log file
- The vmunix and vmcore created by `/etc/savecore`

Kernel Panic Messages 2

accept

File /sys/sys/uipc_syscalls.c

Routine accept

Problem A socket connect queue is empty when sockets should be connected to it.

The socket variable `so_qlen` indicated there were socket connects on the socket connect queue `so_q`, but the routine detected the socket connect queue was empty.

alloc: bad size

File /sys/fs/ufs/ufs_alloc.c

Routine alloc

Problem A file system block being allocated is the wrong size.

When this routine receives a block size, it checks the size of the block before allocating it. In this case, the routine detected the block was either greater than the file system block size or not a multiple of the file system fragment size.

Output Identifies the device from which the block was allocated, the file system block size, the size requested, and the file system. The format is:

```
dev = <0Xd> bsize =<d> size = <d> fs = <"string">
```

alloccg: block not in map

File /sys/fs/ufs/ufs_alloc.c

Routine mapsearch

Problem A cylinder group's free map contains no free blocks.

When allocating a block, the routine searched the file cylinder group summary and found cylinder groups that contain free blocks. However, the search through the free map for one of these cylinder groups detected it contained no free blocks.

Output Identifies the block number and the file system. The format is:

```
bno = <d>  fs = <"string">
```

alloccg: map corrupted

File	/sys/fs/ufs/ufs_alloc.c
Routine	mapsearch
Problem	The cylinder group contains no free fragments. This routine determines whether a requested fragment can be allocated. In this case, the routine searches the free map list and finds a byte that contains free fragments. However, when searching the bits of the byte map to determine which fragment was free, the routine detected there was no free fragment in the cylinder group.
Output	Indicates the starting boundary for the fragment, its length, and the file system. The format is: `start = <d> len = <d> fs = <"string">`

alloccgblk: cant find blk in cyl

File	/sys/fs/ufs/ufs_alloc.c
Routine	alloccgblk
Problem	A free block is not in the free block bit map. The routine found a free block in both the cylinder group table and the file system positional table but could not find the block in the free block bit map.
Output	Identifies the position of the block in the file system positional table, its index, and the file system. The format is: `pos = <d> i = <d> fs = <"string">`

alloccgblk: cyl groups corrupted

File	/sys/fs/ufs/ufs_alloc.c
Routine	alloccgblk
Problem	A free block is not in the file system positional table. The routine found a free block in the cylinder group table but could not find the same free block in the file system positional table.
Output	Identifies the position of the block in the file system positional table, its index, and the file system. The format is: `pos = <d> i = <d> fs = <"string">`

arp Bresolve: no free entry

File	/sys/net/netinet/if_ether.c
Routine	arpresolve
Problem	There are no free entries in the `arp` table and all the entries there are permanent.

auditlog

File	/sys/sys/kern_auditlog.c
Routine	initaud
Problem	`Kmalloc` of space for audit buffer failed.

badaddr

File	/sys/machine/mips/locore.s
Routine	badaddr
Problem	Bad bus address. The routine detected a bus error on a read access to a particular address.

bad c_page

File	/sys/vm/vm_page.c
Routine	checkpage
Problem	A page frame number does not match the page frame number of the clock. When checking for pages to page out, the routine detected that the page frame number of the page currently being checked does not correspond to the `cmap` entry.

bad nofault

File	/sys/machine/mips/trap.c
Routine	trap()
Problem	An exception occurred while the system was processing a previous exception. The system experienced an exception condition (trap) while processing a prior exception and had no way of processing the current exception.

bad mem alloc

File /sys/vm/vm_mem.c

Routine memall

Problem A free memory segment is beyond the bounds of configured physical memory.

This routine allocates physical memory that is represented by core map (cmap) entries. In this case, the routine detected the address of a free memory segment from a cmap entry was beyond the bounds of the configured physical memory of the system.

bad mem free

File /sys/vm/vm_mem.c

Routine memfree

Problem A page frame number is beyond the bounds of configured physical memory.

This routine frees memory. In this case, the routine detected a page table entry page frame number was beyond the bounds of configured physical memory.

bad rmfree

File /sys/sys/subr_rmap.c

Routine rmfree

Problem A resource address or size parameter is invalid.

This routine frees space from a resource map. Before doing so, the routine checks address and size parameters to ensure they do not overlap and are within bounds. The routine detected one of the parameters was invalid because it was out of bounds or overlapped by the other parameter.

big push

File /sys/vm/vm_swp.c

Routine swap

Problem The number of bytes being swapped is greater than the bytes in a software page.

The routine detected the number of bytes it was swapping was greater than the number of bytes in a software page, and the routine was invoked as a consequence of pageout rather than swapout.

blkdev

File /sys/fs/gfs/gfs_bio.c

Routine getblk

Problem The major device number for a block is invalid.

This routine assigns buffers to blocks. Before making the assignment, the routine checks the device number of the device for the block. The routine detected the major device number was out of bounds.

bread

File /sys/fs/gfs/gfs_bio.c

Routine bread

Problem A block is greater than the size of its input buffer.

Before it transfers a block for a buffered read operation, the routine checks the size of the block. The routine detected the size of the block was greater than the input buffer size.

bread: size 0

File /sys/fs/gfs/gfs_bio.c

Routine bread

Problem The size of a block is zero.

Before it transfers a block for a buffered read operation, the routine checks the size of the block. The routine detected the size was zero.

breada

File /sys/fs/gfs/gfs_bio.c

Routine breada

Problem A block is greater than the size of its input buffer.

Before it transfers a block for a buffered read ahead operation, the routine checks the size of the block. The routine detected the size was greater than the input buffer size.

breadrabp

File /sys/fs/gfs/gfs_bio.c

Routine breada

Problem A block is greater than the size of its input buffer.

Before it transfers a block for a buffered read ahead operation, the routine checks the size of the block. The routine detected the size was greater than the input buffer size.

brealloc

File /sys/fs/gfs/gfs_bio.c

Routine brealloc

Problem The space being allocated for a buffer is locked in memory.

While allocating space for a buffer, the routine detected the `B_LOCKED` flag of the buffer was set. When this flag is set, the space for the buffer is locked in memory and cannot be allocated.

brelse: freelist

File /sys/fs/gfs/gfs_bio.c

Routine brelse

Problem A buffer being freed is already free.

Before releasing a buffer to the free list, the routine checks the flag field of the buffer. If this field indicates that the buffer was already marked free, the above panic is issued.

Output The routine issues a message that indicates the buffer pointer, the device, and the gnode pointer and number in the following format:

```
brelse: bp <0Xd> dev <0Xd> gp <0Xd> (<d>) already on list
```

bsc_control

File /sys/net/netbsc/bsc_pcb.c

Routine bsc_control

Problem A pointer to the network interface structure is invalid.

This routine controls bsc operations. In this case, an internet request was received, but the routine detected the request contained a null pointer to the network interface structure.

bsc_usrreq

File /sys/net/netbsc/bsc_usrreq.c

Routine bsc_usrreq

Problem A user request for a bsc operation is invalid.

This routine processes bsc user requests. In this case, the routine received the request but could not recognize the type code of the request.

buffer header allocation failure

File vax - machine/vax/machdep.c

Routine vax - startup()

Problem On a multiprocessor machine, an inconsistent value for memory per buffer header was found, indicating a corrupt kernel image.

buffer header allocation failure

File risc - machine/mips/startup.c

Routine risc - mapinit()

Problem On a multiprocessor machine, an inconsistent value for memory per buffer header was found, indicating a corrupt kernel image.

bufflush pte not valid

File /sys/machine/mips/cache.c

Routine bufflush

Problem Bad page table entry (pte).

While flushing a page from the cache, the routine detected that the page table entry was invalid.

bus timeout

File /sys/machine/mips/trap.c

Routine trap

Problem Bus timeout.

The hardware detected a memory bus error in kernel mode. This panic typically indicates a memory board problem.

bvpdriver: Attempt to open path

File /sys/io/bi/bvp_serv.c

Routine uq_open_path()

Problem The bvp port driver attempted to open a communications path.

The bvp port driver received a request to open a communications path. The driver, however, does not support initiating such connections.

bvp_log_err: Invalid port type

File /sys/io/bi/bvp_subr.c

Routine bvp_log_err()

Problem The system attempted to log an error on a hardware port type not supported by ULTRIX.

bwrite

File /sys/fs/gfs/gfs_bio.c

Routine bwrite

Problem A block is greater than the size of its output buffer.

Before it transfers a block for a buffered write operation, the routine checks the size of the block. The routine detected the block size was greater than the output buffer size.

cbhung

File /sys/io/mba/vax/mba.c

Routine mbintr

Problem The control bus is hung.

The routine cannot process an interrupt from the MASSBUS adapter because the control bus is hung. (This panic is for VAX11/750 processors only.)

Output Identifies the MASSBUS adapter number. The format is:

```
mba <d>: control bus hung
```

character queue overflow

File /sys/sys/kern_clock.c

Routine chrqueue

Problem The console character queue is full.

While checking the console character queue, the routine detected it was full.

checkpage: cmap entry already locked

File sys/vm/vm_page.c

Routine checkpage

Problem A page has changed from unlocked to locked state while the pageout daemon was running. This violates the scheduling protocol required by pageout.

checkpage: invalid swap index

File vm/vm_page.c

Routine checkpage

Problem When checking for pages to page out, checkpage detected the size computed for the segment was greater than the size in the dmap structure of that segment.

checkpage: NULL dmap

File vm/vm_page.c

Routine checkpage

Problem NULL pointer to dmap information.

When attempting to allocate swap space during page out, checkpage detected the segment has a NULL pointer to dmap information.

chkiq

File /sys/fs/gfs/gfs_quota.c

Routine chkiq

Problem The device is not mounted.

Before determining the gnode quota for a mounted device, the routine checks that the device is mounted. The routine detected the device was not mounted.

CHM? in kernel

File /sys/machine/vax/locore.s

Routine kspnotval

Problem A change access mode instruction is invalid.

When the processor detects an instruction that attempts to change access mode from kernel mode to a less privileged mode, it issues an exception and dispatches the exception to this routine. In this case, the routine serviced the exception by producing this panic.

ci - attempting to load unnecessary microcode

File /sys/io/ci/ci_init.c

Routines ci7b_load, cibca_aa_load

Problem The CI port possesses onboard functional microcode.

There are two routines that can issue this panic. The `ci7b_load` routine loads CI7B family functional microcode (CI750/CI780/CIBCI). The `cibca_aa_load` routine loads CIBCA-AA functional microcode.

While performing its function, one of the routines determined that the CI port possessed functional microcode and issued this panic.

ci - attempting to map/unmap already mapped/unmapped adapter

File /sys/io/ci/ci_error.c

Routines ci_map_port, ci_unmap_port

Problem The CI port is already mapped/unmapped.

There are two routines that issue this panic. The `ci_map_port` routine maps CI ports. The `ci_unmap_port` routine unmaps them.

While performing its function, one of the routines determined that either the CI port was already mapped or it was already unmapped.

ci - invalid pccb fork block

File See Table 2-1

Routine See Table 2-1

Problem The necessary pccb data structure is interlocked to prevent use.

There are several CI routines that can issue this panic. These routines perform the various functions that are briefly described in Table 2-1.

Table 2-1: pccb Fork Block Routines

File	Routine	Description
/sys/io/ci/ci_error.c	ci_cleanup_port	Cleans up CI ports.
/sys/io/ci/ci_init.c	ci_init_port	Initializes CI ports. This routine issues a panic when either the data structure necessary for scheduling its asynchronous execution was not interlocked to prevent use or the data structure necessary for scheduling a consecutive asynchronous port initialization attempt is interlocked to prevent use.
	ci_probe	Probes newly discovered CI ports.
/sys/io/ci/ci_isr.c	ci_unmapped_isr	Services interrupts for unmapped CI ports.
/sys/io/ci/ci_lpmaint.c	ci_crash_lport	Crashes CI ports.
/sys/io/ci/cippd_error.c	cippd_clean_fpb	Cleans up formative paths.
/sys/io/ci/cippd_event.c	cippd_stop	Cleans up paths associated with failed ports.
/sys/io/ci/cippd_pmaint.c	cippd_remove_pb	Removes and disposes of path blocks from the Systems Communication Architecture Subsystem database.

ci - invalid unmapping of local port

File /sys/io/ci/ci_isr.c

Routine ci_unmapped_isr

Problem A CI port should not be unmapped.

This routine services interrupts for unmapped CI ports. While processing an interrupt, the routine determined that the port is functional, has power, and should be mapped.

ci - no invalidate translation cache command packet

File /sys/io/ci/ci_subr.c

Routine ci_inv_cache

Problem The reserved port command buffer is absent.

This function invalidates CI port translation caches. Specially reserved port command buffers are used by the routine for invalidating caches as it terminates specific established paths. While performing such an invalidation, the routine discovered the absence of the reserved port command buffer.

ci - no set circuit off command packet

File /sys/io/ci/ci_subr.c

Routine ci_set_circuit

Problem The reserved port command buffer is absent.

This function sets virtual circuits on or off. Specially reserved port command buffers are used by the routine for setting to off circuits associated with specific paths. While setting such a circuit to off, the routine discovered that the reserved port command buffer was absent.

ci - panic requested on all local port failures

File /sys/io/ci/ci_lpmaint.c

Routine ci_crash_lport

Problem A panic was issued based on the setting of the configuration variable `ci_lpc_panic.`

This routine crashes CI ports. While crashing a port, the routine determined that the setting of the CI configuration variable `ci_lpc_panic` (located in ../data/ci_data.c) required that a system panic be issued.

ci - unknown cable status check requested

File /sys/io/ci/ci_subr.c

Routine ci_update_cable

Problem The CI cable-transition check type is unknown.

This routine checks for the existence of a specified type of CI cable transition. While processing a cable, the routine determined that the type of check is unknown.

ci - unknown console logging formatting code

File /sys/io/ci/ci_error.c

Routine ci_console_log

Problem The class of variable information for CI events is unknown.

This routine optionally logs CI events to the console terminal. While logging an event, the routine determined that the class of variable information to be logged is unknown.

ci - unknown interconnect type

File See Table 2-2

Routine See Table 2-2

Problem The CI interconnect type is unknown.

There are several CI routines that can issue this panic. These routines log CI device attention events, map CI ports, and probe newly discovered ports. Table 2-2 briefly explains each of these routines.

Table 2-2: CI Interconnect Routines

File	Routine	Description
/sys/io/ci/ci_error.c	ci_log_dev_attn	Logs CI device attention events.
	ci_map_port	Maps CI ports.
/sys/io/ci/ci_init.c	ci_probe	Probes newly discovered CI ports.

ci - unknown local port crash reason

File /sys/io/ci/ci_lpmaint.c

Routine ci_crash_lport

Problem The CI port is being crashed for an unknown reason.

This routine crashes CI ports. While crashing a CI port, the routine determines the reason for crashing the port is unknown.

ci - unknown/invalid event code

File /sys/io/ci/ci_error.c

Routines ci_console_log, ci_log_initerr

Problem Unknown or invalid CI event code.

There are two routines that can issue this panic. The `ci_console_log` routine logs CI events to the console terminal. While attempting to log an event, the `ci_console_log` routine determined one of the following:

- The event type is unknown.
- The event severity level is invalid or unknown.
- The event is not supposed to be logged by the CI port driver.

The routine `ci_log_initerr` logs CI device attention events that occurred while probing new CI ports. While logging such an event, `ci_log_initerr` determined the event is unknown.

ci - unknown/invalid hardware port type

File — See Table 2-3

Routine — See Table 2-3

Problem — The CI port type is unknown.

There are several CI routines that can issue this panic. These routines handle CI events, interrupts, and various CI port functions. Table 2-3 briefly explains each of these routines.

Table 2-3: Port Checks

File	Routine	Description
/sys/io/ci/ci_error.c	ci_console_log	Logs CI events to the console terminal.
	ci_log_dev_attn	Logs CI device attention events.
	ci7b_disable	Completely disables CI7B family ports (CI750, CI780, and CIBCI).
	cibx_disable	Completely disables CIBX family ports (CIBCA).
/sys/io/ci/ci_init.c	ci_probe	Probes newly discovered CI ports.
	ci_test_port	Checks for the presence of CI ports.
	cibx_start	Starts CIBX family ports (CIBCA).
	ci_unmapped_isr	Sevices interrupts for unmapped CI ports.

ci ppd - broken traffic interval timer

File — /sys/io/ci/cippd_protocol.c

Routines — cippd_start_tmr, cippd_stop_tmr

Problem — The CI PPD traffic interval timer is already started or stopped.

There are two routines that can issue this panic. The `cippd_start_tmr` routine starts the CI PPD traffic interval timer and the `cippd_stop_tmr` routine stops the timer. While performing its function, the routine determined that the timer was previously started or stopped.

ci ppd - invalid path state

File — See Table 2-4

Routine — See Table 2-4

Problem — The CI path is in an invalid state.

The CI routines that can issue this panic are briefly described in Table 2-4.

Table 2-4: Invalid Path Checks

File	Routine	Description
/sys/io/ci/cippd_event.c	cippd_stop	Cleans up paths associated with failed ports.
/sys/io/ci/cippd_protocol.c	cippd_dispatch	Action dispatcher for the CI PPD, finite state machine.
	cippd_enter_db	Enters path blocks into the Systems Communication Architecture Subsystem database.
	cippd_path_schd	Schedules asynchronous cleanup of paths.
	cippd_ppderror	Processes CI PPD protocol violations.
	cippd_rrestart	Processes remote CI PPD path restart requests.

ci ppd - invalid pb fork block

File See Table 2-5

Routine See Table 2-5

Problem The data structure necessary for scheduling asynchronous execution was not interlocked to prevent use.

There are several routines that can issue this panic. All of these routines deal with the cleaning up of pb paths (see Table 2-5 for a brief description).

Table 2-5: Invalid pb Fork Block Routines

File	Routine	Description
/sys/io/ci/cippd_error.c	cippd_clean_fpb	Cleans up formative paths.
	cippd_clean_pb	Cleans up established paths.
/sys/io/ci/cippd_event.c	cippd_stop	Cleans up paths associated with failed ports.
/sys/io/ci/cippd_protocol.c	cippd_path_schd	Schedules asynchronous cleanup of paths.

ci ppd - invalid pccb fork block

File See Table 2-6

Routine See Table 2-6

Problem The necessary ppd data structure is interlocked to prevent use.

There are several CI routines that can issue this panic. These routines perform the various functions that are briefly described in Table 2-6.

Table 2-6: Invalid ppd Fork Block Routines

File	Routine	Description
/sys/io/ci/cippd_error.c	cippd_clean_fpb	Cleans up formative paths.
/sys/io/ci/cippd_event.c	cippd_stop	Cleans up paths associated with failed ports.
/sys/io/ci/cippd_pmaint.c	cippd_remove_pb	Removes and disposes of path blocks from the Systems Communication Architecture Subsystem database.

ci ppd - invalid state or event combination encountered

File /sys/io/ci/cippd_protocol.c

Routine cippd_panic

Problem An unexpected or illegal path state or event combination.

This routine contains unexpected and illegal path state and event combinations. These combinations should never occur in the CI PPD finite state machine.

ci ppd - invalid/unknown path crash reason

File /sys/io/ci/cippd_protocol.c

Routine cippd_pcreason

Problem The CI path-crash event code is unknown. This routine maps a specific path-crash event code into a more general reason for path failure. While mapping such an event code, the routine determined that the event is unknown.

ci ppd - panic requested on all path failures

File /sys/io/ci/cippd_pmaint.c

Routine cippd_crash_pb

Problem The CI PPD configuration variable `cippd_pc_panic` was set.

This function crashes CI PPD paths. While crashing a path, the routine determined that the setting of the PPD configuration variable `cippd_pc_panic` (located in ../data/cippd_data.c) required that a system panic be issued.

ci ppd - path is already enabled

File /sys/io/ci/cippd_protocol.c

Routine cippd_enab_path

Problem The CI PPD path is already enabled.

This function enables CI PPD paths during their establishment. While enabling a path, the routine determined it is already enabled.

ci ppd - removing unremovable path

File /sys/io/ci/cippd_pmaint.c

Routine cippd_remove_pb

Problem The ppd path block cannot be removed from the Systems Communication Architecture Subsystem database.

This routine removes and disposes of path blocks from the Systems Communication Architecture Subsystem database. While processing a path block, the routine determined that the block is not in any condition to be removed.

ci ppd - unknown console logging formatting code

File /sys/io/ci/cippd_error.c

Routine cippd_conlog

Problem The CI PPD variable class is unknown.

This routine optionally logs CI PPD events to the console terminal. While logging an event, the routine determined the class of variable information to be logged is unknown.

ci ppd - unknown finite state machine event

File /sys/io/ci/cippd_protocol.c

Routine cippd_dispatch

Problem Unknown finite state machine event

This function is the action dispatcher for the CI PPD finite state machine. While processing an event, it determined that the event is unknown.

ci ppd - unknown/invalid event code

File /sys/io/ci/cippd_error.c

Routine cippd_conlog

Problem The CI PPD event code is unknown or invalid.

This routine optionally logs CI PPD events to the console terminal. While logging an event, the routine determined the following:

- The event type is unknown.
- The event severity level is invalid.
- The event is unknown.
- The event should not be logged by the CI Port, Port driver.

ci ppd - unknown/invalid system-level event

File /sys/io/ci/cippd_error.c

Routine cippd_csyslev

Problem The PPD, common system-level event is unknown or invalid.

This routine processes CI PPD common system-level events. While processing such an event, the routine determined that it was unknown.

ci ppd - unretrievable path

File /sys/io/ci/cippd_protocol.c

Routine cippd_enter_db

Problem Unable to retrieve a path block for the Systems Communication Architecture Subsystem database.

This function enters path blocks into the Systems Communication Architecture Subsystem database. When the routine is unable to retrieve a block, it issues this panic.

cleanup center

File /sys/vm/vm_page.c

Routine checkpage

Problem There is more than one page kluster associated with a pageout buffer.

While checking a page, the routine detected there was more than one page kluster associated with a pageout buffer.

cleanup CSYS

File /sys/vm/vm_page.c

Routine checkpage

Problem A system page is being paged out.

The routine detected a pageout operation was being performed on a system page. Pageouts must not occur on system pages.

clget: null client

File /sys/fs/nfs/nfs_subr.c

Routine clget

Problem An NFS client structure being allocated cannot be allocated.

This routine sets up client structures for the NFS file system. While doing so, it allocates the structure. In this case, the routine was unable to allocate the structure.

clntkudp_create: kmem_alloc returns 0

File /sys/net/rpc/clnt_kudp.c

Routine clntkudp_create

Problem The system ran out of memory attempting to create a remote procedure call (rpc) handle.

clntkudp_create: kmem_alloc returns 0 for p->cku_outbuf

File /sys/net/rpc/clnt_kudp.c

Routine clntkudp_create

Problem The system ran out of memory attempting to create a remote procedure call (rpc) handle.

closedq: stray dquot

File /sys/fs/gfs/gfs_kernquota.c

Routine closedq

Problem A disk quota structure being removed from a queue is not released.

Before removing a disk quota structure from a file system queue, the reference count of the structure is set to zero to show it is released. When removing a disk quota structure from the file system queue, the routine detected the reference count of the structure was not zero.

clrblock

File /sys/fs/ufs/ufs_subr.c

Routine clrblock

Problem A free block has an invalid number of fragments.

This routine clears a block fragment from the free block map for a cylinder. When it finds a free block, the routine checks the block for the number of file system fragments it contains. In this case, the routine detected the free block had an invalid number of fragments. The number of fragments per block can be only 8, 4, 2, or 1.

coprocessor unusable

File /sys/machine/mips/trap.c

Routine trap()

Problem One of the DECstation coprocessors is not functioning properly.

DECstation systems have more than one coprocessor. The kernel generates this panic if one of these coprocessors is not working properly.

could sleep on interrupt stack

File /sys/sys/kern_lock.c

Routine sleep_check

Problem It is not legal to sleep in an interrupt routine. This is a check to verify that a cpu is in a state where it can reschedule.

could sleep holding spin lock

File /sys/sys/kern_lock.c

Routine sleep_check

Problem It is not legal for a process to be rescheduled while it is holding a spin lock. This is not allowed to avoid a deadlock condition on the spin lock.

DBE not on load or store

File /sys/machine/mips/trap.c

Routine trap()

Problem A data bus error (DBE) occurred that was not a load or store to memory.

A DBE occurred on an instruction that was not performing a read or write operation to memory.

dequeuing non-free text

File sys/vm/vm_text.c

Routine xalloc

Problem There is an attempt to allocate a text structure that has already been allocated.

dequeuing non-free text

File /sys/h/text.h

Routine X_DQFREE {macro definition}

Problem There is an attempt to allocate a text structure that has already been allocated.

This routine dequeues the text structure from the free list.

deuna xmit in progress

File /sys/io/netif/if_de.c

Routine destart

Problem A deuna entry to be transmitted is already being transmitted.

Before transmitting an entry from a transmit buffer queue, the routine first checks the status flag of the entry and then sets the flag to indicate the entry is being transmitted. In this case, the routine checked the flag and discovered it already was set.

dirtysm: no SMS

File	/sys/vm/vax/pt_machdep.c
Routine	dirtysm
Problem	A shared memory segment is not found in a process structure linked to it. This routine checks for modified (dirty) page table entries in shared memory space. When it receives a specific shared memory segment, the routine checks for that segment in a process structure linked to it. In this case, the routine could not find the segment in the process structure.

dirtysm: no SMS #2

File	/sys/vm/vax/pt_machdep.c
Routine	dirtysm
Problem	A shared memory segment is not found in the process structures linked to it. This routine checks for modified (dirty) page table entries in shared memory space. When it receives a specific shared memory segment, the routine checks for that segment in the process structures linked to it. In this case, the routine could not find the segment in any process structure linked to it.

dirtysm: p_sm#

File	/sys/vm/vax/pt_machdep.c
Routine	dirtysm
Problem	The routine detected a process that indicated it had attached shared memory, but the process has a NULL pointer to shared memory information in the process (proc) structure.

dirtysm: smp

File	/sys/vm/vax/pt_machdep.c
Routine	dirtysm
Problem	An offset into a shared memory segment is not a multiple of CLSIZE. This routine checks for modified (dirty) page table entries in shared memory space. When it receives an offset into a shared memory segment, the routine checks the offset to ensure it is a multiple of the system CLSIZE. In this case, the offset parameter was not a multiple of CLSIZE.

discquota

File /sys/fs/gfs/gfs_kernquota.c

Routine dqalloc

Problem A disk quota structure on the free list is not free.

When a disk quota structure is written to the free list, the `DQ_MOD` flag of the structure is cleared to show that the structure is available for reallocation and any modifications to it have been written to disk. While reallocating the disk quota structure, the routine detected the `DQ_MOD` flag was set, indicating the disk quota structure was not free.

distsmpte

File /sys/vm/vax/pt_machdep.c

Routine distsmpte

Problem A shared memory segment is not found in the process structures linked to it.

This routine updates all the page tables of all processes linked to a shared memory segment. When it receives a specific shared memory segment, the routine checks for that segment in the process structures linked to it. In this case, the routine could not find the segment in the process structures linked to it.

distsmpte #2

File /sys/vm/vax/pt_machdep.c

Routine distsmpte

Problem A shared memory segment is not found in a process structure linked to it.

This routine updates all the page table entries of the processes linked to a shared memory segment. When it receives a specific shared memory segment, the routine checks for that segment in a process structure linked to it. In this case, the routine could not find the segment in the process structure.

distsmpte: PG_V && PG_FOD

File /sys/vm/vax/pt_machdep.c

Routine distsmpte

Problem A fill-on-demand page operation is being attempted on a valid page.

While updating the page table entries of the processes linked to a specific shared memory segment, the routine detected it was executing a fill-on-demand page operation on a page already marked as valid.

dli, found_user2

File /usr/src/sys/dli/dli_input.c

Routine dli

Problem Invalid link type.

DLI has found a recipient for a packet, but the link type specified by the calling routine is invalid.

dli_input, found_user2: bad socket

File /usr/projects/wp/sys/net/dli/dli_input.c

Routine found_user()

Problem A socket pointer set to -1 was incorrectly passed to this routine.

dli_bind: eaddr_reserved:

File /usr/src/sys/dli/dli_bind.c

Routine dli_bind

Problem Corrupted DLI line table entry.

While checking for Ethernet address reservations during a bind operation, DLI detected a corruption in one of its line table entries.

dli_bind: osi_ena_802pi:

File /usr/src/sys/dli/dli_subr.c

Routine dli_bind

Problem Corrupted DLI line table entry.

DLI detected a corruption in one of its line table entries while attempting to enable a subnetwork access protocol (SNAP) service access point (SAP) protocol ID.

dli_close

File /sys/net/dli/dli_close.c

Routine dli_close

Problem Corrupted DLI line table entry.

While performing a close operation, DLI detected a corruption in one of its line table entries.

dli_ifoutput

File /usr/src/sys/dli/dli_if.c

Routine dli_ifoutput

Problem Unknown value in the address structure.

While outputting a packet, DLI detected an unknown value in the address structure passed to it by an Ethernet driver.

dli_input: forward_to_user

File /usr/src/sys/dli/dli_input.c

Routine dli_input

Problem Corrupted DLI line table entry.

While searching for the recipient of a packet, DLI detected a corruption in one of its line table entries.

dli_input, found_user1

File /usr/src/sys/dli/dli_input.c

Routine dli_input

Problem Corrupted DLI line table entry.

DLI found a recipient for a packet, but the user's socket pointer is NULL.

dmalloc: bad swap fragment size

File vm/vm_pt.c

Routine dmalloc

Problem Invalid swap fragment value. The global variable that contains the swap fragment value was corrupted as the routine detected a value less than or equal to zero.

dmalloc: bad segsize

File vm/vm_pt.c

Routine dmalloc

Problem An invalid segment size is specified. The routine detected the segment size specified was either less than zero or greater than the system-specified limit.

dmc rcv

File /sys/io/netif/if_dmc.c

Routine dmcxint

Problem There are no buffers available for a DMC11 or DMR11 read operation.

This routine handles interrupts from the DMC11/DMR11 interfaces. When a read interrupt occurs, the routine checks for errors and then notifies the appropriate protocol of the interrupt. In this case, the routine could not find the location of the read buffers in the `dmcuba` structure associated with the interrupt.

dmexpand: NULL dmap

File vm/vm_pt.c

Routine dmexpand

Problem NULL pointer to a dmap structure. Before attempting to expand or contract the dmap structure of a segment, the routine found the segment has a NULL pointer to the dmap information.

dmexpand: bad number of elements

File vm/vm_pt.c

Routine dmexpand

Problem An invalid segment size is specified. The segment size specified was either less than zero or greater than the system-specified limit for that segment.

dmfree: Illegal segtype

File vm/vm_pt.c

Routine dmfree

Problem The segment type cannot be classified as text, data, stack, or shared memory.

dnlc_purge: zero vp

File /sys/fs/nfs/vfs_dnlc.c

Routine dnlc_purge

Problem An entry in the name lookup cache is not associated with a gnode.

The routine detected an entry in the name lookup cache was not associated with a gnode.

dpvread - no mbufs available

File /sys/io/netif/if_dpv.c

Routine dpvread

Problem There are no memory buffers available for a DPV11 read operation.

This routine handles input from the DPV11. While getting a memory buffer to hold the input, the routine detected there were no memory buffers available.

dup biodone

File /sys/fs/gfs/gfs_bio.c

Routine biodone

Problem A block being transferred is already transferred.

This routine marks a block I/O as completed. Before doing so, the routine checks the `B_DONE` flag in the flags field of the block's buffer structure. Then, it sets the flag to mark the block as being transferred. While checking the flag, the routine detected it was already set.

dup mem alloc

File /sys/vm/vm_mem.c

Routine memall

Problem A page on the free list is not free.

This routine allocates physical memory by core map (cmap) entries. While doing so, the routine checks the cmap entry to ensure it is marked free. The routine detected a page on the free list was not marked free.

dup mem free

File /sys/vm/vm_mem.c

Routine memfree

Problem A page cluster being freed is already free.

This routine frees physical memory by core map (cmap) entries. While doing so, the routine checks the cmap entry to ensure it is not already free. The routine detected a page cluster that was already marked free.

evl_usrreq

File /usr/src/decnet/evl/evl_krtns.c

Routine evl_usrreq

Problem Illegal user request on an event logger (EVL) socket.

exec: EFAULT

File /sys/sys/kern_exec.c

Routine execve

Problem There is an argument error while executing a child process.

This routine executes a new process on top of itself. To do so, the routine copies the calling process's arguments to temporary storage. Later, these arguments are copied back to user address space. While copying the arguments back, the routine detected there was an argument error or some discrepancy between the arguments received and those copied back.

exit

File /sys/sys/kern_exit.c

Routine exit

Problem A process structure is not in the pid hash table.

This routine terminates processes. To do so, the routine locates the process structure for the process in the process identification (pid) hash table. In this case, the routine did not find the process structure for the process in the hash table.

expand

File /sys/vm/vm_proc.c

Routine expand

Problem A request to change P0/P1 space is not a multiple of CLSIZE.

This routine changes the size of the data or stack regions for a process. Before doing so, the routine checks the size requested, to ensure it is a multiple of CLSIZE. The routine detected the requested size was not a multiple of CLSIZE.

expand

File /sys/vm/vm_proc.c

Routine smexpand

Problem A request to resize P0 space to map or unmap a shared memory segment is not a multiple of CLSIZE.

This routine changes the size of P0 space to map/unmap a shared memory segment for a process. Before doing so, the routine checks the size to ensure it is a multiple of CLSIZE. The routine detected the request was not a multiple of CLSIZE.

fhandle and lockhandle-id are not the same size!

File /sys/fs/nfs/nfs_vnodeops.c

Routine nfs_rlock

Problem An inconsistency has been detected in the Network File System (NFS) file locking.

fifo_open: KM_ALLOC

File /sys/fs/specfs/fifo_gnodeops.c

Routine fifo_open

Problem The system ran out of memory attempting to allocate space for a fifo structure.

This routine is called to open both pipes (fifos) and named pipes. It attempts to allocate space for a fifo structure that hangs off the fifo's gnode.

flushpte: !isasms

File /sys/machine/mips/vm_machdep.c

Routine flushpte

Problem Shared memory data structure could not be located.

While flushing the translation lookaside buffer (tlb) of the passed-in shared memory pages, the routine could not locate the associated per-process shared memory data structure for one of the virtual pages of the faulting process.

flushpte: smindex == -1

File /sys/machine/mips/vm_machdep.c

Routine flushpte

Problem Shared memory data structure could not be located

While flushing the translation lookaside buffer (tlb) of the passed-in shared memory pages, the routine could not locate the associated per-process shared memory data structure for one of the attached sharing processes.

fodkluster

File /sys/vm/vm_page.c

Routine fodkluster

Problem There is not enough memory to allocate for a page kluster.

This routine finds adjacent pages for pagein and pageout operations. When it finds the pages, the routine checks that it has enough free memory to allocate a page kluster, prior to allocating them. However, the `memall` routine returned indicating that there is not enough memory.

free: bad size

File /sys/fs/ufs/ufs_alloc.c

Routine free

Problem A block being freed is the wrong size.

This routine receives the size of a block or fragment to free. While attempting to free the block or fragment, the routine detected it was either greater than the file system block size or not a multiple of the file system fragment size.

Output Identifies the device from which the block was freed, the file system block size, the size requested, and the file system. The format is:

```
dev = <0Xd> bsize = <d> size = <d> fs = <"string">
```

free_cpu: invalid cause

File /sys/sys/kern_cpu.c

Routine free_cpu

Problem An undefined reason to restart a cpu was sent to the routine.

free: freeing free block

File /sys/fs/ufs/ufs_alloc.c

Routine free

Problem A block being freed is already free.

This routine receives the block number of a block or fragment to free. While attempting to free the block or fragment, the routine detected it was already in the free block map.

Output Identifies the device from which the block was freed, the block number, and the file system. The format is:

```
dev = <0Xd> block = <d> fs = <"string">
```

free: freeing free frag

File /sys/fs/ufs/ufs_alloc.c

Routine free

Problem A fragment being freed is already free.

This routine receives the block number of a fragment to free. While attempting to free the fragment, the routine detected it was already in the free block map.

Output Identifies the device that contains the block, the block number, and the file system. The format is:

```
dev = <0Xd> block = <d> fs = <"string">
```

freegnode: freeing active gnode

File /sys/fs/gfs/gfs_gnodeops.c

Routine freegnode

Problem A gnode being freed is active.

This routine frees gnodes when they are no longer active. While doing so, the routine detected the reference count for the gnode was not zero, indicating it was still active.

Output Indicates the routine and the gnode address and number. The format is:

```
freegnode: gp <0Xd> (<d>)
```

Freeing free text

File /sys/h/text.h

Routine X_QFREE {macro definition}

Problem There is an attempt to free a text structure that has already been freed.

freeing gnode already on free list

File /sys/fs/gfs/gfs_gnodeops.c

Routine freegnode

Problem A gnode being freed is already free.

This routine frees gnodes. In this case, the routine checked the gnode free list and detected the gnode was already free.

Output Indicates that NFS is inactive and the gnode address and number. The format is:

```
nfs_inactive: gp <0Xd> (<d>)
```

fstat

File /sys/fs/gfs/gfs_descrip.c

Routine fstat

Problem The file type field of a file descriptor is invalid.

This routine checks the status of a file. While doing so, the routine detected the descriptor's file type field was invalid because it did not equal the value for the `inode`, `socket`, or `port` variable.

gap_accept

File /usr/src/sys/ccitt/gap_usrreq.c

Routine gap_usrreq

Problem Missing pointer to the sockaddr structure.

A server has passed a sockaddr structure to the kernel and the structure was corrupted.

gap_send

File /usr/src/sys/ccitt/gap_usrreq.c

Routine gap_usrreq

Problem Missing control block of the connected socket.

When a connection between the gap server and the application is established, a control block exists for each socket of the socket pair. However, this control block is missing when an application attempts a send call.

gap_sendoob

File /usr/src/sys/ccitt/gap_usrreq.c

Routine gap_usrreq

Problem Missing control block of the connected socket.

When a connection between the gap server and the application is established, a control block exists for each socket of the socket pair. However this control block is missing when an application attempts a send call for `oob` data.

geteblk: zero length buffer

File /sys/fs/gfs/gfs_bio.c

Routine geteblk

Problem A block's size is zero.

This routine gets empty blocks for later assignment to devices. In this case, the routine detected the block it obtained was invalid because its size was zero.

getegnode: free gnode isnt

File /sys/fs/gfs/gfs_gnodeops.c

Routine getegnode

Problem A gnode on the free list is still active.

This routine gets a gnode from the free list and checks the gnode's reference count, which is zero when a gnode is not active. In this case, the routine detected the reference count was not zero, indicating the gnode was still active.

Output Indicates the routine and the gnode address and number. The format is:

```
getegnode: gp <0Xd> (<d>)
```

gfs_unlock: locked gnode, no unlock routine

File /sys/fs/gfs/gfs_gnode.c

Routine gfs_unlock

Problem There is no unlocking routine for a locked gnode.

This routine unlocks gnodes. Some gnodes do not have an unlocking routine and, therefore, should never be locked. In this case, the routine checked the type of the locked gnode and detected there was no unlocking routine for it.

gfs_unlock: unlocked gnode

File /sys/fs/gfs/gfs_gnode.c

Routine gfs_unlock

Problem A gnode being unlocked is already unlocked.

This routine unlocks gnodes. Before unlocking a gnode, the routine checks the gnode structure. While doing so, the routine detected the gnode was already unlocked.

Output Identifies the routine, the gnode address and number, and the device. The format is:

```
gfs_unlock: gp <0Xd> (<d>) dev <0Xd>
```

gno_lock

File /sys/fs/gfs/gfs_gnodeops.c

Routine gno_lock

Problem A gnode is not released from a shared or an exclusive lock.

This routine places an advisory lock on a gnode. Before it can do so, any shared or exclusive locks on the gnode must be released. When the routine discovers a gnode that has such a lock, it sleeps until the gnode is released. In this case, the routine detected the shared or advisory lock, slept, woke up, and then detected the gnode was still locked.

gno_unlock: EXLOCK

File /sys/fs/gfs/gfs_gnodeops.c

Routine gno_unlock

Problem The lock type of the file is exclusive but the gnode is not exclusive.

This routine unlocks files. Before it does so, the routine checks the file lock type and the file's gnode flags. In this case, the routine detected the file lock type was exclusive, but the gnode flags indicated the file was not locked exclusive.

gno_unlock: SHLOCK

File /sys/fs/gfs/gfs_gnodeops.c

Routine gno_unlock

Problem The lock type of a file is shared but the gnode is not shared.

This routine unlocks files. Before it does so, the routine checks the file lock type and the file's gnode flags. In this case, the routine detected the file lock type was shared, but the gnode flags indicated the gnode was not shared.

got bad quota uid

File /sys/fs/gfs/gfs_kernquota.c

Routine getquota

Problem The disk quota for a user id has no match in its disk quota structure.

After receiving a user id, the routine compares the user id's quota against the same user id's quota in the disk quota structure. The routine detected the values did not match.

gput g_count < 1!

File /sys/fs/gfs/gfs_gnode.c

Routine gput

Problem The reference count of a gnode structure is invalid.

While trying to decrement the reference count of a gnode structure, the routine detected the reference count was invalid because it was less than one.

Output Indicates the routine, the gnode address, the device address, and gnode number. The format is:

```
gput: gp <0Xd> g_dev <0Xd> number <d>
```

gvp - illegal buffer name

File /sys/io/gvp/gvp_block.c

Routine gvp_unmap_buf

Problem The VAXport buffer descriptor is either invalid or it cannot be authenticated.

hard IO err in swap

File /sys/vm/vm_swp.c

Routine swap

Problem There is a hard I/O error during a swap out request.

The routine detected there was a hard I/O error while it was performing a swap I/O operation. This error usually indicates a hardware problem.

hardclock: p_sm

File /sys/sys/kern_clock.c

Routine hardclock

Problem The routine detected a process that indicated it had attached shared memory, but the process has a NULL pointer to shared memory information in the process (proc) structure.

hold_cpu: invalid cause

File /sys/sys/kern_cpu.c

Routine hold_cpu

Problem An undefined reason to idle a cpu was sent to the routine.

hold_cpu: on primary

File /sys/sys/kern_cpu.c

Routine hold_cpu

Problem Boot cpu cannot be put into idle state.

holding lock on trap exit

File /sys/machine/vax/trap.c

Routine trap

Problem Process is still holding an SMP lock trying to return to user mode. This indicates a missed unlock in the kernel.

holding lock on syscall exit

File /sys/machine/vax/trap.c

Routine syscall

Problem Process is still holding an SMP lock trying to return to user mode. This indicates a missed unlock in the kernel.

holding lock on syscall exit

File /sys/machine/mips/trap.c

Routine syscall

Problem Process is still holding an SMP lock trying to return to user mode. This indicates a missed unlock in the kernel.

hpsize: invalid partition table

File /sys/io/mba/vax/hp.c

Routine hpsize

Problem The disk's partition table is invalid.

Before determining the size in blocks of a partition, the routine checks the disk's partition table. While checking the partition table, the routine detected it was invalid.

hpstrategy: invalid partition table

File /sys/io/mba/vax/hp.c

Routine hpstrategy

Problem The disk's partition table is invalid.

Before it queues a disk read or write request, the routine first checks the disk's partition table and then retrieves a block number. While checking the partition table, the routine detected it was invalid.

ialloccg: block not in map

File /sys/fs/ufs/ufs_alloc.c

Routine ialloccg

Problem There is a free gnode in a cylinder, but none in the gnode map.

This routine determines whether a requested gnode can be allocated. In this case, the routine determines the requested gnode is not available and checks each cylinder in the cylinder group for an unused gnode. When it finds a cylinder that indicates it has an unused gnode, the routine finds the location for that gnode by searching the used gnode map for that cylinder. The routine detected the used gnode map did not contain an unused gnode for the cylinder.

Output Indicates the file system. The format is:

```
fs = <"string">
```

ialloccg: map corrupted

File /sys/fs/ufs/ufs_alloc.c

Routine ialloccg

Problem There are free gnodes in the cylinder group, but none in the gnode map.

This routine determines whether a requested gnode can be allocated. In this case, the routine determines the requested gnode is not available, but the free gnode count for the cylinder group indicates there are other gnodes available. Next, the routine attempts to locate the next unused gnode in the cylinder group by scanning the used gnode map for the cylinder group. The routine detected there were no available gnodes in the used gnode map, but the free gnode count for the cylinder group indicated there were unused gnodes available.

icmp len

File /sys/net/netinet/ip_icmp.c

Routine icmp_error

Problem Bad internet control message protocol (icmp) packet.

The packet exceeded the size of the memory buffer (mbuf), or the packet got corrupted while it was being examined.

icmp_error

File /sys/net/netinet/ip_icmp.c

Routine icmp_error

Problem A message type code in an impc header is invalid.

The routine detected the message type code in an impc header did not match one of the predefined types.

idcsize: invalid partition table

File /sys/io/uba/idc.c

Routine idcsize

Problem The disk's partition table is invalid.

Before it determines the size (in blocks) of a partition, the routine first checks the disk's partition table. While checking the partition table, the routine detected it was invalid.

idcstrategy: invalid partition table

File /sys/io/uba/idc.c

Routine idcstrategy

Problem The disk's partition table is invalid.

Before it queues a disk read or write request, the routine first checks the disk's partition table and then retrieves a block number. While checking the partition table, the routine detected it was invalid.

in_control

File /sys/net/netinet/in.c

Routine in_control

Problem A pointer to the network interface structure is invalid.

This routine handles internet control operations. In this case, the routine received a request but the request included a null pointer to the network interface structure.

init died

File /sys/sys/kern_exit.c

Routine exit

Problem The `init` process is being terminated.

This routine terminates processes. Before doing so, it locates the process structure for the exiting process in the process identification (pid) hash table. In this case, the routine detected the pid for the process was that of the init process.

init_main: cdir == NULL

File /sys/sys/init_main.c

Routine main

Problem The gnode of the current directory is lost.

This routine retrieves the gnode of the current directory. In this case, a NULL pointer was returned to the routine, instead of the gnode pointer.

init_main: rootdir == NULL

File /sys/sys/init_main.c

Routine main

Problem The gnode of the root directory is lost.

This routine retrieves the gnode of the root directory. In this case, the routine received a NULL pointer instead of the gnode pointer.

inoquota

File /sys/fs/gfs/gfs_quota.c

Routine inoquota

Problem The device is not mounted.

The routine receives a gnode pointer. Because there is no in-memory disk quota structure associated with this gnode pointer, the routine attempts to look up a matching gnode variable in the mount structure. The routine detected there was not a valid device associated with the gnode structure, so the device was not mounted.

intrpt_cpu: invalid cpu

File /sys/sys/kern_cpu.c

Routine intrpt_cpu

Problem A processor tried to send an interprocessor interrupt to a cpu that does not exist.

invalid cylinder

File /sys/io/uba/sdc.c

Routine sdstart

Problem The disk cylinder number is invalid.

After calculating the disk cylinder number, the routine detected it was greater than the number of cylinders on the disk.

Output Identifies the device, the device unit number, and the invalid cylinder number. The format is:

```
device:<d> unit:<d> :HARD ERROR: Invalid cylinder:<d>
```

invalid head

File /sys/io/uba/sdc.c

Routine sdstart

Problem A disk head number is invalid.

After calculating the disk head number, the routine detected it was greater than the number of heads on the disk.

Output Identifies the device, the device unit number and the invalid head number. The format is:

```
device:<d> unit:<d> :HARD ERROR: Invalid head:<d>
```

IO err in push

File /sys/vm/vm_swp.c

Routine swdone

Problem A hard I/O error occurs as a page kluster is being transferred.

The routine detected a hard I/O error had occurred as a page kluster was being transferred. This error typically indicates a hardware problem.

ip_init

File /sys/net/netinet/ip_input.c

Routine ip_init

Problem An entry for a protocol family cannot be put into the protocol switch table.

This routine puts entries for protocol families into the internet protocol switch table. In this case, the routine was unable to put an entry for a protocol family into the switch table.

isblock

File /sys/fs/ufs/ufs_subr.c

Routine isblock

Problem A free block has an invalid number of fragments.

When it finds a free block, the routine checks the block for the number of fragments it contains. Then, the routine compares that number to the number of fragments allowed by the file system. In this case, the routine detected the free block had an invalid number of fragments. The number of fragments per block can be only 8, 4, 2, or 1.

kern_audit: no mem

File	/sys/sys/kern_audit.c
Routine	audit_rec_build
Problem	Kmalloc of space for audit buffer failed.

kernel used coprocessor

File	/sys/machine/mips/locore.s
Routine	tbs
Problem	A floating point operation was attempted in kernel mode.

kluster

File	/sys/vm/vm_page.c
Routine	kluster
Problem	There is not enough free memory to allocate a page kluster.
	This routine finds adjacent pages for pagein and pageout operations. Prior to allocating memory for the kluster, the routine checks to ensure that there is enough free memory. However, the `memall` routine returns indicating that there is not enough free memory.

KM_ALLOC: bucket corruption

File	/sys/vm/vm_kmalloc.c
Routine	KM_ALLOC {macro definition}
Problem	The address of the next piece of memory to be allocated is outside the virtual address space controlled by the kernel memory allocator.

km_alloc: bucket corruption

File	/sys/vm/vm_kmalloc.c
Routine	km_alloc
Problem	The address of the next piece of memory to be allocated is outside the virtual address space controlled by the kernel memory allocator.

KM_FREE: bad addr

File	/sys/vm/vm_kmalloc.c
Routine	KM_FREE {macro definition}
Problem	The address passed into KM_FREE is not a system address.

km_free: bad addr

File /sys/vm/vm_mem.c

Routine km_free

Problem An address passed into the KM_FREE routine is outside the virtual address space controlled by the kernel memory allocator.

km_free: bad index

File /sys/vm/vm_kmalloc.c

Routine km_free

Problem The segment being freed has a resource list index that is out of bounds.

KM_FREE: multiple frees

File /sys/h/kmalloc.h

Routine KM_FREE {macro definition}

Problem The number of references to the segment being freed has been decremented to a negative value, or the number of segments available has been incremented above the number possible.

km_free: multiple frees

File /sys/vm/vm_mem.c

Routine km_free

Problem The number of references to a segment being freed has been decremented to a negative value, or the number of segments available has been incremented above the number possible.

km_memdup not a cluster

File /sys/vm/vm_kmalloc.c

Routine km_memdup

Problem An illegal operation attempted to raise the reference count of segments smaller than a page cluster.

kpteseg

File /sys/machine/mips/trap.c

Routine tlbmiss

Problem The virtual address is not in user space.

While servicing a virtual address in KPTESEG space, the routine detected that the virtual address was not in user space as expected, but in kernel space.

kpteseg miss outside utlbmiss

File /sys/machine/mips/trap.c

Routine tlbmiss

Problem Table missing

While servicing a virtual address in KPTESEG space, the routine determined that the routine was not entered as a result of a `utlbmiss`.

Output The following output indicates the program counter at the time of the exception (epc) and the faulting virtual address that is being serviced (vaddr). Note that epc must be within the `utlbmiss` handler.

```
epc= 0x%x, vaddr= 0x%x
```

KSP not valid

File /sys/machine/vax/locore.s

Routine kspnotval

Problem The kernel stack pointer is invalid.

When the processor detects the kernel stack pointer is outside the bounds of the kernel stack, it issues an exception and dispatches the error to this routine. In this case, the routine serviced the exception by producing this panic.

lat_usrreq

File /sys/net/lat/lat_usrreq.c

Routine lat_usrreq

Problem A user request for a lat operation is invalid.

This routine processes lat user requests. In this case, the routine received the request but could not recognize the type code of the request.

ldctx

File /sys/machine/vax/locore.s

Routine _Resume

Problem The process context cannot be restored.

This routine restores context for a process that had been blocked. In this case, the routine attempted to restore the context but failed to do so.

longjump

File /sys/machine/vax/locore.s

Routine _Longjump

Problem The kernel stack frame pointer is invalid.

This routine saves various parameters, and then checks them prior to a jump operation. In this case, the routine detected the stack frame pointer it had saved was not the same as the current stack frame pointer.

lost quota file

File /sys/fs/gfs/gfs_kernquota.c

Routine putdq

Problem There is no valid quota file pointer in the mount structure.

When there are quotas on a file system, there is a pointer to the quota file in the file system mount structure. While attempting to update or free a disk quota, the routine detected there was no valid quota file pointer in the mount structure.

lost shared memory

File /sys/vm/vm_smem.c

Routine smunlink

Problem A process sharing a memory segment is not in the list of sharing processes for that segment.

This routine removes processes from a list of processes sharing a shared memory segment. While doing so, the routine could not find the process in the list of processes sharing the memory segment.

lost text

File /sys/vm/vm_text.c

Routine xunlink

Problem A process sharing a text segment is not in the segment's list of processes.

This routine removes a process from the list of those processes sharing a text segment. While doing so, the routine detected the address of the process was not in the shared text segment's list of processes.

main: can't init dmap

File init_main.c

Routine main

Problem Unable to allocate kernel memory for dmap structure for data segment.

main: can't init smap

File init_main.c

Routine main

Problem Unable to allocate kernel memory for dmap structure for stack segment.

maknode: dquot

File /sys/fs/ufs/ufs_syscalls.c

Routine maknode

Problem A free gnode is not free.

After it receives a free gnode for a new file, the routine checks the gnode's disk quota pointer. When the pointer is zero, the gnode is not associated with a disk quota structure and is free. In this case, the routine detected the gnode's disk quota pointer was not zero. The gnode was associated with a disk quota structure and was not free.

mapin

File /sys/machine/mips/vm_machdep.c

Routine mapin

Problem The page table entry (pte) does not reside in KSEG2 space.

While dropping a page table entry into the translation lookaside buffer (tlb), the routine detected that the virtual address the pte maps does not reside in KSEG2 space.

maunhash ecmap

File /sys/vm/vm_mem.c

Routine maunhash

Problem Memory hash chain corruption.

The routine failed to find the core map entry, even though the calling routine validated that the entry exists. This error usually indicates a memory hash chain corruption.

maunhash: mfind

File /sys/vm/vm_mem.c

Routine maunhash

Problem The routine detected a second instance of a hashed core map (cmap) entry.

A cmap entry can appear only once among the text memory hash chains.

maunhash: unhashing non text page

File /sys/vm/vm_mem.c

Routine maunhash

Problem The cmap entry to be unhashed has been discovered to have a type that is not CTEXT.

mba, zero entry

File /sys/io/mba/vax/mba.c

Routine mbasetup

Problem A page table contains invalid entries.

The routine sets up MASSBUS adapter map registers from page table entries. The delimiter for the page table is a page table entry that contains a zero in its page frame bits. While searching the page table, the routine detected a page table entry with zero in its page frame bits before it found the delimiter for the page table.

mbintr

File /sys/io/mba/vax/mba.c

Routine mbintr

Problem The device driver returns an invalid instruction.

This routine receives an interrupt from the MASSBUS adapter and determines the interrupt is a data transfer operation. Then, the routine calls the device driver to process the operation. After the driver returned an instruction, the routine detected it was invalid.

mbintr

File /sys/io/mba/vax/mba.c

Routine mbintr

Problem The device driver returns an invalid instruction.

This routine receives an interrupt from the MASSBUS adapter and determines the interrupt is a nondata transfer operation. Then, the routine calls the device driver to process the operation. After the driver returned an instruction, the routine detected it was invalid.

mbustart

File /sys/io/mba/vax/mba.c

Routine mbustart

Problem The device driver returns an invalid status code or instruction.

This routine calls the unit start routine for a MASSBUS device. The device driver then returns a status code or an instruction. The routine detected the status code or instruction returned from the driver is invalid.

mchk

File See Table 2-7

Routine See Table 2-7

Problem The operating system cannot recover from a machine check.

There are several routines that handle machine checks issued by processors. These routines determine whether the operating system can recover from the machine check issued. Table 2-7 contains the routines that handle machine checks.

Output Identifies the machine check type code and other diagnostic information. See the *Guide to System Crash Recovery* for more information.

Table 2-7: Machine Checks

File	Routine	Processors
/sys/machine/vax/ka610.c	ka610machcheck	MicroVAX I
/sys/machine/vax/ka6200.c	ka620machcheck	VAX 62xx or VAX 63xx series
/sys/machine/vax/ka630.c	ka630machckeck	MicroVAX II
/sys/machine/vax/ka6400.c	ka6400machcheck	VAX 8400 series
/sys/machine/vax/ka730.c	ka730machckeck	VAX 730
/sys/machine/vax/ka750.c	ka750machcheck	VAX 750
/sys/machine/vax/ka780.c	ka780machcheck	VAX 780/785
/sys/machine/vax/ka8200.c	ka8200machcheck	VAX 8200/8300/8250/8350
/sys/machine/vax/ka8600.c	ka8600machcheck	VAX 8600/8650
/sys/machine/vax/ka8800.c	ka8800machcheck	VAX 8500/8550/8700/88xx series

mcldup has bad m_cltype

File /sys/sys/uipc_mbuf.c

Routine mcldup

Problem A memory buffer being duplicated has an invalid cluster type.

This routine duplicates memory buffers according to the cluster type associated with them. In this case, the routine checked the cluster type and detected it was invalid because it was an unknown type.

m_copy1

File /sys/sys/uipc_mbuf.c

Routine m_copy

Problem A memory buffer has a negative length or offset parameter.

Thisroutine is passed several parameters that it uses to copy memory buffers for use with sockets. In this case, the routine discovered that either the length or offset parameter passed to it was less than zero.

m_copy2

File /sys/sys/uipc_mbuf.c

Routine m_copy

Problem A memory buffer pointer points to the end of the memory buffer chain.

This routine is passed several parameters that it uses to copy memory buffers for use with sockets. The routine detected the memory buffer pointer parameter passed to it was a null pointer.

m_copy2 got a bad mbuf

File /sys/net/rpc/subr_kudp.c

Routine m_copy2

Problem The type code of a memory buffer in a chain is invalid.

This routine detected a memory buffer in a chain had a type code other than 2, or the offset into the buffer was greater than the buffer size.

m_copy3

File /sys/sys/uipc_mbuf.c

Routine m_copy

Problem The copy operation for a memory buffer is invalid.

This routine is passed several parameters that it uses to copy memory. The routine detected the length parameter passed to it did not equal a `copyall` operation, which is the only valid operation when the memory buffer pointer value is zero.

mda0: zero pfn in pte

File /sys/machine/vax/md.c

Routine mdstrategy

Problem A page table contains invalid entries.

This routine maps memory from page table entries. The delimiter for the page table is a page table entry that contains zero in its page frame bits. While searching the page table, the routine detected a page table entry containing zero in its page frame bits before the delimiter for the page table.

memall

File /sys/vm/vm_mem.c

Routine memall

Problem The size of the memory being allocated is not a multiple of CLSIZE.

This routine allocates physical memory. After it receives the size of the memory it is to allocate, the routine checks its size to ensure it is a multiple of CLSIZE. The routine detected the size of the memory it was allocating was not a multiple of CLSIZE.

memall intrans|want

File /sys/vm/vm_mem.c

Routine memall

Problem A free cluster is marked as in-transit or wanted by another process.

This routine allocates physical memory. While doing so, the routine gets free page clusters from core map (cmap) entries and checks the cmap flags to ensure the flag settings match the cluster state. In this case, the routine detected the in-transit or wanted flags were set in the cmap. The cluster is not free.

memerr

File /sys/machine/vax/ka8800.c

Routine ka8800memerr

Problem A VAX 8800 memory error.

This routine determines whether memory errors detected by the KA-8800 processor are recoverable. In this case, the routine decided it was not possible to recover from the memory error.

Output Identifies the problem. The format is:

```
VAX 8800 memory error
```

memfree

File /sys/vm/vm_mem.c

Routine memfree

Problem The size of memory being freed is not a multiple of CLSIZE.

This routine frees physical memory. While doing so, the routine detected the size of the memory being freed was not a multiple of CLSIZE.

memfree: freeing intrans|want page

File sys/vm/vm_mem.c

Routine memfree

Problem The system has detected that a page being freed is in use by some other thread of execution.

memintr, memory failure

File /sys/machine/mips/trap.c

Routine trap()

Problem A write to a valid physical memory address failed.

This error indicates a hardware failure.

memintr, write timeout

File /sys/machine/mips/trap.c

Routine trap()

Problem The system tried to write to an invalid (nonexistent) address.

This error could indicate either a hardware or a software failure.

memory parity error in kernel mode

File /sys/machine/mips/trap.c

Routine trap()

Problem A memory parity error occurred while the system was running in kernel mode.

The system has no way to recover from this error and consequently shuts down.

memory parity error in shared page

File /sys/machine/mips/trap.c

Routine trap()

Problem A memory parity error occurred while the system was running in shared mode.

A memory parity error occurred in a shared page while the system was running in user mode. The system does not have enough information to terminate all processes that were sharing the page and consequently shuts down.

mfind: trying to find non text on hash

File /sys/vm/vm_mem.c

Routine mfind

Problem A core map (cmap) entry on a hash chain has been discovered to have a type that is not CTEXT.

m_free has bad m_cltype

File /sys/h/mbuf.h

Routine mbuf

Problem An illegal or corrupted memory (mbuf) has been freed.

This panic usually indicates that a process has overwritten the control part of a memory buffer. When the mbuf is freed, the routine found that this type of mbuf is not valid.

mhash: no mp

File /sys/vm/vm_mem.c

Routine mhash

Problem There is no mount point for the specifed device.

This routine adds physical text pages from a specified device to the hash chain. In this case, the routine detected there was no mount point for the specified device.

mkdir: blksize

File /sys/fs/ufs/ufs_syscalls.c

Routine ufs_mkdir

Problem The DIRBLKSIZ system parameter is greater than the file system fragment size.

This routine checks critical system parameters, such as DIRBLKSIZ, to monitor their size. While monitoring the DIRBLKSIZ system parameter, the routine detected its size was greater than the file system fragment size.

mkdir: dquot

File /sys/fs/ufs/ufs_syscalls.c

Routine mkdir

Problem A free gnode is not free.

After it receives a free gnode for a new directory, this routine checks the gnode's disk quota pointer. When the pointer is zero, the gnode is not associated with a disk quota structure and is free. In this case, the routine detected the gnode's disk quota pointer was not zero. The gnode was associated with a disk quota structure and was not free.

mmrw

File /sys/machine/vax/mem.c

Routine mmrw

Problem There is memory left to read or write, but the I/O count is zero.

This routine processes memory addresses for reads or writes to physical or virtual memory. As a read or write operation is completed, the routine decrements the I/O count. When the I/O vector count is zero, the routine compares it against the residual memory count variable. The routine detected the I/O count was zero, but the residual memory count was greater than zero.

mount_root: can't kmem_alloc. . .

File /sys/data/gfs_data.c

Routine mount_root:

Problem The system ran out of memory trying to allocate space for the device name string.

This routine is called to mount both a local root device and a root device on a remote host for a diskless machine.

mountrpc: cannot NFS mount file

File /sys/fs/nfs/nfs_vfsops.c

Routine mountrpc

Problem Client did not recieve a valid response from server.

The mount deamon (mountd) on the server did not respond sucessfully for the mount request of the client root device.

mscp_alloc_msg: double msg buffer allocation

File /sys/io/sysap/mscp_subr.c

Routine mscp_alloc_msg

Problem An attempt was made to allocate an MSCP message buffer for a request that already holds one.

mscp_alloc_rspid: double RSPID allocatio

File /sys/io/sysap/mscp_subr.c

Routine mscp_alloc_rspid

Problem An attempt was made to allocate a response ID (RSPID) for a request that already holds one.

mscp_bbr_force: no BBR work area allocated

File /sys/io/sysap/mscp_bbr.c

Routine mscp_bbr_force

Problem The bbr work area pointer in the connection block is zero

This bbr work area pointer is filled in by the `mscp_bbr_init` routine. The connection block or the request block has been corrupted.

mscp_bbr_init: couldn't allocate BBRB

File /sys/io/sysap/mscp_bbr.c

Routine mscp_bbr_init

Problem The bad block replacement (bbr) work area could not be allocated from memory.

This routine allocates and initializes a work area for bbr operations. This panic is generated when the `KM_ALLOC` routine fails to allocate memory for the bbr work area.

mscp_bbr_lock: no BBR work area allocated

File /sys/io/sysap/mscp_bbr.c

Routine mscp_bbr_lock

Problem The bbr work area pointer in the connection block is zero.

This bbr work area pointer is filled in by the `mscp_bbr_init` routine. The connection block is corrupted or the request block has been corrupted.

mscp_concleanup: disconnect failed

File /sys/io/sysap/mscp_conpol.c

Routine mscp_concleanup

Problem The `scs_disconnect` routine returned a connection busy error.

mscp_concomplete: connect to an active server

File /sys/io/sysap/mscp_conpol.c

Routine mscp_concomplete

Problem A connect was issued to an already connected server.

This error indicates an error in the connection management finite state machine.

mscp_concomplete: unrecognized status

File /sys/io/sysap/mscp_conpol.c

Routine mscp_concomplete

Problem The `scs_connect` routine returned an unrecognized completion status.

mscp_conqrestart: duplicate sequence numbers

File /sys/io/sysap/mscp_conpol.c

Routine mscp_conqrestart

Problem Two requests with the same sequence number were queued to the connection restart queue, or the same request was queued twice.

mscp_conresynch: scs_reset failed

File /sys/io/sysap/mscp_conpol.c

Routine mscp_conresynch

Problem Either the `scs_reset` routine found the port in an unexpected state, or it could not allocate the resources needed to restart the port.

mscp_conresynch: scs_restart failed

File	/sys/io/sysap/mscp_conpol.c
Routine	mscp_conresynch
Problem	Either the `scs_restart` routine found the port in an unexpected state, or it could not allocate the resources needed to restart the port.

mscp_constconem: no credit available

File	/sys/io/sysap/mscp_conpol.c
Routine	mscp_constconem
Problem	No controller credits available.
	Insufficient credits were available for the class driver to send a set controller characteristics message.

mscp_control: unexpected connection management event

File	/sys/io/sysap/mscp_subr.c
Routine	mscp_control
Problem	SCS called the class driver with an unrecognized event code.

mscp_dealloc_msg: bad connection state or ID

File	/sys/io/sysap/mscp_subr.c
Routine	mscp_dealloc_msg
Problem	Either the SCS connection was in an improper state to process the deallocate request (for example, when disconnect has been completed) or the connection ID passed to SCS was invalid.

mscp_dealloc_rspid: sequence number mismatch

File	/sys/io/sysap/mscp_subr.c
Routine	mscp_dealloc_rspid
Problem	The sequence number portion of the response ID did not match the sequence number in the RSPID table.

mscp_invevent: fatal mscp error

File /sys/io/sysap/mscp_subr.c

Routine mscp_invevent

Problem Invalid event.

One of the finite state machines used in processing requests detected an event that was not valid in the current state. This panic is preceded by a message that identifies the event, the state, and the address of the request block in error in the following format:

Output

```
mscp_invevent:  invalid event <d> in state <d>, reqb <x>
```

mscp_map_buffer: double buffer handle allocation

File /sys/io/sysap/mscp_subr.c

Routine mscp_map_buffer

Problem An attempt was made to allocate an MSCP buffer handle for a request that already holds one.

mscp_message: invalid rspid

File /sys/io/sysap/mscp_subr.c

Routine mscp_message

Problem Response ID in the MSCP end message did not match the RSPID in the corresponding request block.

This panic is preceded by a message that displays the mismatching RSPIDs in the following format:

Output

```
mscp_message: end msg rspid <x> != rp rspid <x>
```

mscp_message: scs_dealloc_msg failed

File /sys/io/sysap/mscp_subr.c

Routine mscp_message

Problem The `scs_dealloc_msg` routine returned an error indicating that the connection was in an inappropriate state or that the connection ID was invalid.

mscp_recycle_rspid: sequence number mismatch

File /sys/io/sysap/mscp_subr.c

Routine mscp_recycle_rspid

Problem The sequence number portion of the response ID did not match the sequence number in the RSPID table.

mscp_size: invalid partition table

File /sys/io/sysap/mscp_disk.c

Routine mscp_size

Problem The specified partition has been opened, but the partition table information for the device is not valid.

mscp_strategy: invalid partition table

File /sys/io/sysap/mscp_disk.c

Routine mscp_strategy

Problem The specified partition has been opened, but the partition table information for the device is not valid.

mscp_unmap_buffer: bad connection state or ID

File /sys/io/sysap/mscp_subr.c

Routine mscp_unmap_buffer

Problem Either the SCS connection was in an improper state to process the unmap request (for example, when disconnect has been completed), or the connection ID passed to SCS was invalid.

MSG_UNLOCK

File /sys/h/msg.h

Routine MSG_UNLOCK {macro definition}

Problem This routine was called on an unlocked message (msg) queue data structure.

msi - broken transmit fork process timer

File /sys/io/msi/msi_isr.c

Routine msi_xfp_timer

Problem This routine delays transmission of specific packets under specific circumstances from local MSI ports. During the process of delaying specific packet transmissions, the routine determined its interval timer is broken.

msi - invalid pccb fork block

File See Table 2-8

Routine See Table 2-8

Problem There are several routines that can issue this panic. These routines perform the functions that are briefly described in Table 2-8.

Table 2-8: Invalid pccb Fork Block Routines

File	Routine	Description
/sys/io/msi/msi_error.c	msi_clean_port	Cleans up local MSI ports following their failure. While cleaning up of a specific local MSI port, the `msi_clean_port` routine discovers irregularities in the scheduling of its clean up.
/sys/io/msi/msi_init.c	msi_init_port	Initializes local MSI ports. While initializing a specific local MSI port, the `msi_init_port` routine discovers irregularities in the scheduling of its initialization.
	msi_probe	Probes newly discovered local MSI ports. While probing a newly discovered local MSI port, the `msi_probe` routine discovers irregularities preventing the scheduling of port initialization.
/sys/io/msi/msi_lpmaint.c	msi_crash_lport	Crashes local MSI ports. While crashing a local MSI port, the `msi_crash_lport` routine discovers irregularities preventing the scheduling of port clean up.

msi - invalid receive fork process fork block

File /sys/io/msi/msi_isr.c

Routine msi_rfp

Problem Processes all packets received by local MSI ports. While processing packets received by a specific local MSI port, the routine found irregularities in the scheduling of packet processing.

msi - invalid transmit fork process fork block

File /sys/io/msi/msi_isr.c

Routine msi_xfp

Problem Processes all packets transmitted from local MSI ports. While processing packets transmitted from a specific local MSI port, the routine found irregularities in the scheduling of packet processing.

msi - invalid transmit fork process retdat packet

File /sys/io/msi/msi_isr.c

Routine msi_xfp

Problem Processes all packets transmitted from local MSI ports. While processing a packet transmitted from a specific local MSI port, the routine found irregularities in the internal driver processing of a write request.

msi - panic requested on all local port failures

File /sys/io/msi/msi_lpmaint.c

Routine msi_crash_lport

Problem Crashes local MSI ports. While crashing a specific local MSI port, the routine determined that the setting of the MSI configuration variable `msi_lpc_panic` (located in ../data/msi_data.c) required that a system panic be issued.

msi - unknown console logging formatting code

File /sys/io/msi/msi_error.c

Routine msi_console_log

Problem Optionally logs MSI events to the console terminal. While logging an event, the routine determined that the class of variable information to be logged is unknown.

msi - unknown/invalid event code

File See Table 2-9

Routine See Table 2-9

Problem There are several routines that can issue this panic. These routines perform the functions that are briefly described in Table 2-9.

Table 2-9: Unknown or Invalid Event Code Routines

File	Routine	Description
/sys/io/msi/msi_error.c	msi_console_log	Optionally logs MSI events to the console terminal. While logging an event, the `msi_console_log` routine determined one of the following: the event type is unknown, the event is an MSI-specific event, the event severity level is invalid, the event is unknown to the MSI port driver, or the event is not supposed to be logged by the MSI port driver.

Table 2-9: (continued)

File	Routine	Description
	msi_log_initerr	Logs those MSI device attention events occurring during probing of new local MSI ports. While logging an event, the `msi_log_initerr` routine determined the event is unknown to the MSI port driver.

msi - unknown/invalid local port crash reason

File /sys/io/msi/msi_lpmaint.c

Routine msi_crash_lport

Problem Crashes local MSI ports. While crashing a local MSI port, the routine determined that the port being crashed is either unknown or invalid.

muhash: unhashing non text page 2

File /sys/vm/vm_mem.c

Routine munhash

Problem A core map (cmap) entry on a hash chain has been discovered to have a type that is not CTEXT.

munhash

File /sys/vm/vm_mem.c

Routine munhash

Problem A core map (cmap) entry cannot be found in the hash chain.

This routine removes cmap entries for specified devices and block numbers from hash chains and is called only if the `mfind` routine determined that the entry resides on the hash chains. In this case, the routine could not find a cmap entry on the hash chains.

munhash: ecmap

File /sys/vm/vm_mem.c

Routine munhash

Problem A core map (cmap) entry cannot be found in the hash chains.

This routine removes cmap entries for specified devices and block numbers from hash chains. In this case, the routine could not find the cmap entry in the hash chains.

munhash mfind

File /sys/vm/vm_mem.c

Routine munhash

Problem A core map (cmap) entry removed from the hash chain is still on the chain.

This routine removes cmap entries for specified devices and block numbers from the hash chains. After it removes the entries, the routine rechecks the hash chains. In this case, the routine detected an additional entry which indicates hash list corruption.

munhash: unhashing non text page

File /sys/vm/vm_mem.c

Routine munhash

Problem A core map (cmap) entry on a hash chain has been discovered to have a type that is not CTEXT.

MUNLOCK: dup page unlock

File /sys/h/vmmac.h

Routine MUNLOCK {macro definition}

Problem The core map (cmap) entry is already unlocked.

This routine unlocks cmap entries. This panic is issued when the routine is called to unlock an entry that is already unlocked.

newproc: alloc p_sm

File /sys/sys/kern_fork.c

Routine newproc

Problem The kernel memory allocator failed to allocate memory for the shared memory information of the child process (a km_alloc() problem).

newproc: parent has smem, smseg == 0

File /sys/sys/kern_fork.c

Routine newproc

Problem The parent process in a fork has non-NULL shared memory information, although the system was configured without shared memory.

This panic usually indicates process structure corruption.

newproc vfork

File /sys/sys/kern_fork.c

Routine newproc

Problem A child process is swapped out when its parent is awakened.

This routine creates child processes. After it creates the child process, the routine waits until the child completes processing before waking the parent process. This panic is issued when the routine detects the child process is swapped out when the parent awakens. This prevents the parent from being able to reclaim its resources.

nfs_badop

File /sys/fs/nfs/nfs_vnodeops.c

Routine nfs_badop

Problem The generic file system (gfs) has performed an erroneous file system call.

nfs_lock: unrefed gnode

File /sys/fs/nfs/nfs_gfsops.c

Routine nfs_lock

Problem The reference count of a gnode structure is invalid.

This routine unlocks a gnode. When it receives the gnode, the routine checks the reference count in the gnode structure. In this case, the routine detected the reference count in the gnode structure was invalid because it was equal to or less than zero.

nfs_mountrpc cannot get port for mount service

File /sys/fs/nfs/nfs_vfsops.c

Routine mountrpc

Problem Client could not connect to the mountd of the server.

This routine calls the mount deamon (mountd) of the server to verify and return a file handle for the mount point on the server. When the client attempted to obtain the port number the mountd of the server is listening on, an error occurred. The requested mount point was the root device or the client.

nfs_rele: zero count

File /sys/fs/nfs/nfs_gfsops.c

Routine nfs_rele

Problem The reference count for a gnode is invalid.

This routine releases gnodes. While doing so, the routine checks the gnode reference count. In this case, the routine detected the reference count for the gnode was invalid because it was less than or equal to zero.

nfs_unlock: locked gnode isnt

File /sys/fs/nfs/nfs_gfsops.c

Routine nfs_unlock

Problem A gnode being unlocked is already unlocked.

This routine unlocks gnodes. Before unlocking a gnode, the routine checks the gnode structure. While doing so, the routine detected the gnode was already unlocked.

nmi fault

File /sys/machine/vax/ka8800.c

Routine ka8800nmifault

Problem A VAX 8800 NMI fault results in a fatal adapter error.

This routine detects nbia/nbib adapter errors that result from a VAX 8800 NMI (Nautilus memory interconnect) fault. Then, the routine decides whether it is possible to recover from the adapter errors. In this case, the routine decided it was not possible to recover from the adapter error.

nml_ifioctl

File /usr/src/decnet/k_nml/nml_ifioctl.c

Routine nml_ifioctl

Problem This routine was passed an invalid pointer to the interface of a network driver.

no access to shared text

File /usr/src/sys/vax/trap.c

Routine trap

Problem A page table entry (pte) for shared text (executable image) does not permit read access.

no bus data

File /sys/io/xmi/xmiinit.c

Routine get_xmi

Problem The linked list of XMI data structures has been corrupted.

No cachenbl routine configured

File /sys/machine/vax/machdep.c

Routine cachenbl

Problem The cache-enabling function cannot be found during configuration.

During configuration, processor-specific functions are enabled through the `cpusw` data structure in the `cpuconf.c` file. Among these functions is one to enable the cache of the processor. When calling a routine to enable the cache, the routine detected an invalid return. Either the wrong processor routine is called or the cache-enabling routine is not configured.

no CCA

File /sys/machine/vax/cvax.c

Routine cca_setup

Problem No console communication area was found.

No configuration routine configured

File sys/machine/mips/machdep.c

Routine configure()

Problem The processor-specific routine to handle I/O configuration cannot be found through the cpusw structure.

During configuration, processor-specific functions are enabled through the cpusw data structure in the `cpuconfig.c`file. Among these functions is one to configure I/O devices. When calling through the cpusw structure to the I/O configuration routine the routine detected an invalid return. Either the wrong processor routine is called, or the processor routine to handle I/O configuration is not configured.

No cons_putc routine configured

File /sys/machine/vax/machdep.c

Routine cons_putc

Problem The logical console routine cannot be found during configuration.

During configuration, processor-specific functions are enabled through the `cupsw` data structure in the `cpuconf.c` file. Among these functions is one to enable the processor routine for sending instructions to the logical console. While calling a routine to enable instructions to the logical console, the routine detected an invalid return. Either the wrong processor routine is called, or the logical console routine of the processor is not configured.

no hard error interrup routine configured

File sys/machine/mips/trap.c

Routine memintr()

Problem The processor-specific routine handle system hard errors interrupts cannot be found through the cpusw structure.

During through the cpusw data structure in the `cpuconf.c` file. Among these functions is one to handle system hard error interrupts. When calling through the cpusw structure to stop the TOD clock, the routine detected an invalid return. Either the wrong processor routine is called or the processor hard error interrupt handler is not configured.

No harderr_intr handler configured

File /sys/machine/vax/errlog.c

Routine logsbi

Problem The harderr_intr handler function cannot be found during configuration.

During configuration, processor-specific functions are enabled through the cpusw data structure in the cpuconfig.c file. Among these functions is one to enable error handlers for bus errors. When calling a routine to handle a bus error, the routine detected an invalid return. Either the wrong processor routine was called or the processor bus error handler routine is not configured.

No machine check handler configured

File /sys/machine/vax/machdep.c

Routine machinecheck

Problem The machine check handler function cannot be found during configuration.

During configuration, processor-specific functions are enabled through the `cpusq` data structure in the `cpuconf.c` file. Among these functions is one to enable the processor machine check handler. When calling a routine to enable the processor machine check handler, the routine detected an invalid return. Either the wrong processor routine is called or the processor machine check handler is not configured.

no mbufs available

File /sys/io/netif/if_dpv.c

Routine dpvread

Problem There are no memory buffers available for system data structures.

This routine handles input from the DPV11. While getting a memory buffer to hold the input, the routine detected there were no memory buffers available.

no mbufs available

File /sys/io/netif/if_dup.c

Routine dupread

Problem There are no memory buffers available for a DUP11 read operation.

This routine handles input from the DUP11. While getting a memory buffer to hold the input, the routine detected there were no memory buffers available.

no mem for probe i/o

File /sys/machine/vax/autoconf.c

Routine unifind

Problem There is not enough memory to probe UNIBUS I/O space.

When the system is bootstrapped, the UNIBUS is probed for existing devices. Part of the probe is to allocate the first page of UNIBUS I/O space to the first page of memory. The routine detected there was not enough memory available to allocate the first page of UNIBUS I/O space.

no mem for unifind

File /sys/machine/vax/autoconf.c

Routine unifind

Problem There is not enough memory to map UNIBUS I/O space.

The routine allocates memory for UNIBUS I/O space. While allocating memory, the routine detected there was not enough memory available for the memory map.

no memory (A)

File /sys/machine/vax/machdep.c

Routine mapinit

Problem No memory for user mode processes.

At system startup time, the routine allocates space for all the kernel data structures. After performing these allocations, the routine discovered there is no memory space available to start any user mode processes.

no memory (B)

File /sys/machine/vax/machdep.c

Routine mapinit

Problem No memory for user mode processes.

At system startup time, the routine allocates space for all the kernel data structures. After performing these allocations, the routine discovered there is no memory space available to start any user mode processes.

no procs

File /sys/sys/kern_fork.c

Routine newproc

Problem There are no free process slots for a new process.

This routine creates a new process. After `fork`, the calling routine for this routine, has determined there is a free process slot, this routine detected there was no free process slot.

No read TOD routine configured

File sys/machine/mips/machdep.c

Routine read_todclk()

Problem The processor-specific routine to read the TOD clock cannot be found through the cpusw structure.

During configuration, processor-specific functions are enabled through the cpusw data structure in the `cpuconfig.c` file. Among these functions is one to read the TOD clock. When calling through the cpusw structure to read the TOD, the routine detected an invalid return. Either the wrong processor routine is called, or the processor routine to read the TOD clock is not configured.

No setcache routine configured

File /sys/machine/vax/machdep.c

Routine setcache

Problem The function to set the state of the cache cannot be found during configuration.

During configuration, processor-specific functions are enabled through the `cpusw` data structure in the `cpuconf.c` file. Among these functions is one to set the state of the cache of the processor. While calling a routine to set the processor cache, the routine detected an invalid return. Either the wrong processor routine is called or the processor routine to set the cache is not configured.

No softerr_intr handler configured

File /sys/machine/vax/machdep.c

Routine memerr

Problem The memory error-handling function cannot be found during configuration.

During configuration, processor-specific functions are enabled through the `cpusw` data structure in the `cpuconf.c` file. Among these functions is one to enable the processor memory error handler. While calling a routine to enable the handler, the routine detected an invalid return. Either the wrong processor routine is called or the processor memory error handler is not configured.

No start clock routine configured

File sys/machine/mips/machdep.c

Routine startrtclock()

Problem The processor-specific routine to start the TOD clock cannot be found through the cpusw structure.

During configuration, processor-specific functions are enabled through the cpusw data structure in the `cpuconf.c` file. Among these functions is one to start the TOD clock. When calling through the cpusw structure to start the TOD clock, the routine detected an invalid return. Either the wrong processor routine is called or the processor start clock routine is not configured.

No stop clock routine configured

File sys/machine/mips/machdep.c

Routine stopclocks()

Problem The processor-specific routine to stop the TOD clock cannot be found through the cpusw structure.

During configuration, processor-specific functions are enabled through the cpusw data structure in the `cpuconf.c` file. Among these functions is one to stop the TOD clock. When calling through the cpusw structure to stop the TOD clock, the routine detected an invalid return. Either the wrong processor routine is called or the processor stop clock routine is not configured.

No timer_action routine configured

File /sys/machine/vax/machdep.c

Routine memenable

Problem The memory controller function for CRD errors cannot be found during configuration.

During configuration, processor-specific functions are enabled through the `cpusw` data structure in the `cpuconf.c` file. Among these functions is one to report corrected data errors (CRD errors) from memory controllers. When calling a routine to enable memory controller CRD error reporting, the routine detected an invalid return. Either the wrong processor routine is called or the processor controller error-reporting routine is not configured.

no trap error routine configured

File sys/machine/mips/trap.c

Routine trap()

Problem The processor-specific routine to handle an error cannot be found through the cpusw structure.

During configuration, processor-specific functions are enabled through the cpusw data structure in the `cpuconf.c` file. Among these functions is one to handle trap errors. When calling through the cpusw structure to the trap error handler the routine detected an invalid return. Either the wrong processor routine is not configured.

no vector

File /sys/io/bi/biinit.c

Routine bisetvec

Problem No error vector was created for a BI bus that was in the system configuration file.

no vector

File /sys/io/xmi/xmiinit.c

Routine xmisetvec

Problem No error vector was created for an XMI bus that was in the system configuration file.

No write TOD routine configured

File sys/machine/mips/machdep.c

Routine write_todclk()

Problem The processor-specific routine to write the TOD clock cannot be found through the cpusw structure.

During configuration, processor-specific functions are enabled through the cpusw data structure in the `cpuconf.c` file. Among these functions is one to write the TOD clock. When calling through the cpusw structure to write the TOD, the routine detected an invalid return. Either the wrong processor routine is called or the processor routine to write the TOD clock is not configured.

noncontig alloc in qe

File /sys/io/netif/if_qe.c

Routine qeprobe

Problem No contiguous memory found while probing the device.

This error can be caused by lack of system memory or by an unsupported device added to the Q-BUS that used up all the available memory.

not bootcpu

File /sys/sys/kern_clock.c

Routine chrqueue, intqueue

Problem A non-boot processor tried to run code that is only to be used by the boot processor for console character handling.

obreak: p_sm

File /sys/sys/kern_mman.c

Routine obreak

Problem The routine detected a process that indicated it had attached shared memory, but the process has a NULL pointer to shared memory information in the proc structure.

pagein: Attempt to pagein kernel/user shared memory page

File /sys/vm/vm_page.c

Routine pagein

Problem An attempt was made to page-in a shared memory page that is currently being shared between a user process and the system.

Because physical pages reside in system space that should be locked, this panic is fatal.

pagein: bncache=-1

File vm/vm_page.c

Routine pagein

Problem This routine handles page faults. Before attempting to load the page from swap disk, the routine found that the swap space was not allocated for that page.

pagein: bn=-1

File vm/vm_page.c

Routine pagein

Problem Before attempting to bring the page from swap disk, the routine found that the swap space was not allocated for that page.

pagein c_page chgd

File /sys/vm/vm_page.c

Routine pagein

Problem A core map (cmap) entry page number does not match the formatting page number.

This routine handles page faults. In this case, the routine compares the page number, retrieved from the memory hash chains, with the page number derived from the retrieved cmap entry. The routine detected that the pages did not match. This error indicates a memory hash list corruption.

pagein intrans|want

File /sys/vm/vm_page.c

Routine pagein

Problem A free text page is marked as in-transit or wanted by another process.

This routine handles page faults. While handling a fault for a text page, the routine retrieves the core map (cmap) entry from the memory hash chains and checks the cmap flags to ensure the flag settings match the text page state. In this case, the routine detected the in-transit or wanted flags were set in the cmap and the text page was not free.

pagein mfind

File /sys/vm/vm_page.c

Routine pagein

Problem A text page being reclaimed is not free.

This routine handles page faults. While handling a fault, the routine retrieves a text page from the free list and verifies that the page it receives is free. This panic is issued when the routine detects that the page is owned by some other process, the page is not free, or the page is not a text page.

pagein: no process in context

File sys/vm/vm_page.c

Routine pagein

Problem A page fault has occurred when no user process is in context. Since the page fault handler can only resolve user addresses, this is a fatal error.

pagein pfnum

File /sys/vm/vm_page.c

Routine pagein

Problem A page table entry (pte) has a pointer to a page frame number.

This routine handles page faults. In this case, the routine requires a page that is not a fill-on-demand page and that a page frame is not currently assigned to that pte.

Subsequently, the routine checks the original state of the page to ensure it does not contain a page frame number. During this check, the routine detected the original page table entry contained a page frame number and therefore was either a fill-on-demand page in memory or a reclaimable page that should have been validated at this point.

pagein: pfnum = 0

File /sys/vm/vm_page.c

Routine pagein

Problem The page frame number is not allocated.

This routine handles page faults. By this point in processing, the physical page should have been allocated and the page frame number inserted in the page table entry. However, the routine found that the page frame number field in the page table entry is null.

pagein pg_fileno

File /sys/vm/vm_page.c

Routine pagein

Problem A fill-on-demand page has an unknown type code.

This routine handles page faults. While handling a fill-on-demand fault, the routine receives a file number and checks the page table to ensure that the page is a fill-on-demand page (one fetched through the TEXT map). This panic is issued when the routine detects the page had an unknown type code (for example, CTEXT, CDATA, CSTACK, and CSMEM).

pagein PG_FTEXT

File /sys/vm/vm_page.c

Routine pagein

Problem A file is mapped but its text structure is missing.

This routine handles page faults. In this case, the routine receives a file number indicating a fill-on-demand page is mapped from the `a.out` file. However, the routine cannot find the associated text structure.

pagein: SHMEM fodkluster

File /sys/vm/vm_page.c

Routine pagein

Problem A page fault is attempting to fill-on-demand kluster a shared memory page.

This routine handles page faults. While doing so, the routine detected a request to kluster a fill-on-demand page from a shared memory segment. Shared memory segments are exclusively zero fill-on-demand.

pagein SMEM

File /sys/vm/vm_page.c

Routine pagein

Problem A shared memory table entry cannot be found for the given virtual address.

This routine handles page faults. While handling a page fault in shared memory, the routine searches for the shared memory segment structure that contains the page. During this search, the routine failed to find the applicable data structure.

pagein vread

File /sys/vm/vm_page.c

Routine pagein

Problem The page fault option is not supported by the operating system.

After a page fault has occurred, the routine detected an association between a page table entry and a file descriptor. This association is not supported by ULTRIX.

pagein: vtod

File vm/vm_page.c

Routine pagein

Problem The page is not in swap device. This routine handles page faults. This case specifies, load the page from swap space. However, the routine found that the swap space was not allocated for that page.

pagemove

File /sys/machine/mips/vm_machdep.c

Routine pagemove

Problem The size of the data being moved is not a multiple of the page cluster size.

While attempting to move data from one virtual address to another, the routine detected the size of the data was not a multiple of the ULTRIX page cluster size (1024 bytes).

pagemove

File /sys/machine/vax/vm_machdep.c

Routine pagemove

Problem The size of the data being moved is not a multiple of the page cluster size.

While attempting to move data from one virtual address to another, the routine detected the size of the data was not a multiple of the ULTRIX page cluster size (1024 bytes).

pageout: checkpage

File /sys/vm/vm_page.c

Routine checkpage

Problem A valid page table entry (pte) does not contain a valid page frame number (pfnum).

While checking a page to page out, the routine found a pte with the valid bit set and a zero pfnum.

pageout klsize

File /sys/vm/vm_page.c

Routine checkpage

Problem A page has a kluster size less than zero.

While checking a page, the routine detected the kluster size of the page was less than zero.

pagin: p_sm

File /sys/vm/vm_page.c

Routine pagein

Problem NULL pointer to shared memory information.

The routine detected a process that indicated it had attached shared memory, but the process has a NULL pointer to shared memory information in the process (proc) structure.

pagin SMEM

File /sys/vm/vm_drum.c

Routine vtod

Problem A shared memory segment cannot be found.

This routine converts virtual page numbers to disk block numbers. While doing so, the routine failed to find a shared memory segment for a specified shared memory virtual address.

pfalloc: bad mem alloc

File /sys/vm/vm_mem.c

Routine pfalloc

Problem A free memory segment is beyond the bounds of configured physical memory.

This routine allocates page clusters from core map (cmap) entries. In this case, the routine detected the index into a cmap was greater than the size of configured physical memory.

pfalloc: dup mem alloc

File /sys/vm/vm_mem.c

Routine pfalloc

Problem A page on the free list is not free.

This routine allocates physical memory for kernel use. While allocating a physical page, the routine checks the newly allocated core map entries to ensure they are marked free. The routine detected a page on the free list that was not marked free.

pfalloc: intrans|want

File /sys/vm/vm_mem.c

Routine pfalloc

Problem A free page cluster is marked in-transit or wanted by another process.

This routine allocates physical memory for use by the kernel. To do so, the routine allocates physical memory by the core map (cmap) entries and checks the cmap flags to ensure the flag settings match the cluster state. In this case, the routine detected that the in-transit or wanted flags were set in the cmap. Therefore, the cluster is not free.

pfalloc: type

File /sys/vm/vm_mem.c

Routine pfalloc

Problem A page cluster being allocated has an invalid type code.

This routine allocates physical memory for use by the kernel. While doing so, the routine validates that the input parameter type is equal to the constent `CSYS`. This panic is issued when these values are not equal.

pffree: bad mem free

File /sys/vm/vm_mem.c

Routine pffree

Problem A page frame number is beyond the bounds of configured memory.

This routine frees physical page klusters by placing them on the cmap free list. While doing so, the routine detected that the first page frame number was either equal to 1, or that a page frame number was out of the range of configured physical memory.

pffree: dup mem free

File /sys/vm/vm_mem.c

Routine pffree

Problem A page cluster being freed is already free.

This routine frees physical page clusters by placing them on the core map free list. While doing so, the routine detected a physical page was already marked as free.

PfiltAllocatePacket

File /sys/net/net/pfilt.c

Routine PfiltAllocatePacket

Problem While attempting to allocate an Ethernet packet from the free list, the list was found to be empty.

PfiltDeallocatePacket: refcount != 0

File /sys/net/net/pfilt.c

Routine PfiltDeallocatePacket

Problem While attempting to place a deallocated Ethernet packet back onto the free packet queue, the reference count was found to be non-zero.

Pfiltwmove: uio_iovcnt < 0 while uio_resid > 0

File /sys/net/net/pfilt.c

Routine Pfiltwmove

Problem While attempting to transfer the contents of a user buffer into kernel mbufs, a size mismatch was discovered.

pfilt_attach: not enough memory

File /sys/net/net/pfilt.c

Routine pfilt_attach

Problem While attempting to attach to the packet filter, a call to allocate kernel memory for descriptor buffers failed.

pfilt_newaddress: bad unit number

File /sys/net/net/pfilt.c

Routine pfilt_newaddress

Problem While attempting to change the Ethernet hardware address, an invalid interface unit number was specified.

PfiltDeWaitQueue

File /sys/net/net/pfilt.c

Routine PfiltDeWaitQueue

Problem While attempting to remove an Ethernet packet from the wait queue, the queue was found to be empty.

probe i/o space not at bus virtual address 0

File /sys/machine/vax/autoconf.c

Routine unifind

Problem The first page of UNIBUS I/O space is mapped to physical address zero (0).

When the system is bootstrapped, this routine allocates and initializes the UBA map registers and buffered data paths. Part of this process is to allocate the first page of UNIBUS I/O space to the first available page of memory. The routine detected the first page of UNIBUS I/O space had been mapped to physical address zero and not to the first available page.

proc_del: not alive state

File /sys/sys/kern_psubr.c

Routine proc_del

Problem A process is not in P_ALIVE state when trying to move to P_DYING state. This indicates a bug in the process exit code.

proc_del: bad ref

File /sys/sys/kern_psubr.c

Routine proc_del

Problem Once a process is in the P_DYING state, the code waits for all outstanding references to this process to be cleared before proceeding. This panic indicates that the reference count is not zero after waiting.

processor type not configured

File sys/machine/vax/machdep.c, sys/machine/mips/startup/c

Routine cpuswitch_entry

Problem The cpuswitch table is searched for the processor type at boot time. A global pointer to the cpuswitch entry for this processor type is set up to allow a fast index into the cpuswitch table. If a cpuswitch entry cannot be found in the cpuswitch table for this processor type, the kernel cannot continue to run.

Boot the generic kernel (genvmunix) and determine if this processor type is in the configuration file used to build the kernel that panicked. If the processor type for this system is not in the config file for this kernel, add this processor type to the configuration file.

proc_rele: bad ref

File /sys/sys/kern_psubr.c

Routine proc_rele

Problem A process reference count went negative. This indicates that a process was released twice.

proc_exit: not dying state

File /sys/sys/kern_psubr.c

Routine proc_exit

Problem A process is not in P_DYING state when trying to move to P_DEAD state. This indicates a bug in process exit code.

proc_exit: holding a lock

File /sys/sys/kern_psubr.c

Routine proc_exit

Problem A process has completed exit, but still holds an smp lock.

proc_wait: child alive state

File /sys/sys/kern_psubr.c

Routine proc_wait

Problem The routine proc_wait should only be called when waiting for a process to go from the P_DYING state to the P_DEAD state.

psig

File /sys/sys/kern_sig.c

Routine psig

Problem A process has a signal, but no signal flag.

This routine performs signal actions passed to it by processes that have signals. In this case, the routine was called by a process to perform the action, but detected the process did not have a signal flagged in its signal bits.

psig action

File /sys/sys/kern_sig.c

Routine psig

Problem A process requests a signal action, but the action is to ignore the signal.

This routine processes an action specified by a signal. In this case, the action specified is to ignore the signal, which should have been intercepted before being passed to this routine.

ptable fault

File /sys/machine/vax/trap.c

Routine trap

Problem A page table fault results in a processor trap.

This routine detects and handles traps issued by processors. In this case, the routine detected a processor trap caused by a page fault on a page-mapping page table. ULTRIX does not support paging page tables.

ptcmp: No matching ioctl address in block device table

File /sys/fs/ufs/ufs_xxx.c

Routine ptcmp

Problem A matching `ioctl` address for a raw device cannot be found.

This routine matches `ioctl` addresses for raw devices by searching the block device table. In this case, the routine traversed the table but could not find an `ioctl` block device to match the raw device.

ptcwrite

File /sys/sys/tty_pty.c

Routine ptcwrite

Problem There is data to write (to a pseudoterminal), but no I/O vectors contain data.

The I/O count in the I/O structure indicated there were no vectors holding data, but the I/O structure of the user contained a value that indicated there was more data to be written.

ptexpand

File /sys/vm/mips/pt_machdep.c

Routine ptexpand

Problem A request to expand a page table is not a multiple of CLSIZE.

This routine expands a page table. In this case, the routine detected the request to expand the page table was not a multiple of CLSIZE or was less than or equal to zero.

ptexpand

File /sys/vm/vax/pt_machdep.c

Routine ptexpand

Problem A request to expand a page table is not a multiple of CLSIZE.

This routine expands a page table. In this case, the routine detected the request to expand the page table was not a multiple of CLSIZE or was less than or equal to zero.

ptrace regmap botch

File /sys/sys/sys_process.c

Routine procxmt

Problem Illegal register width.

The data structure that contains the width of the register was corrupted as the routine detected a value other than 1, 2, or 4.

Qbus Adapter Dump Error

File sys/machine/vax/ka60.c

Routine ka60memerr()

Problem An error occurred while a QBUS device was writing to main memory. Part or all of the data that was written by the QBUS device did not make it to main memory.

qe: chained packet

File /sys/io/netif/if_qe.c

Routine qerint

Problem An input packet is being chained.

The routine detected an input packet was being chained.

qe: Non existent memory interrupt

File /sys/io/netif/if_qe.c

Routine qeintr

Problem An Ethernet interrupt occurs, but the memory for it does not exist.

This routine processes Ethernet interrupts. When it receives an interrupt, the routine checks the control and status flags of the register. The routine detected the flag for nonexistent memory was set.

raw_usrreq

File /sys/net/net/raw_usrreq.c

Routine raw_usrreq

Problem The protocol type code of a user request is invalid.

The routine receives a user request. In this case, the request was invalid because the routine detected the protocol type code of the request was not one of the predefined types in the raw protocol switch table.

realloccg: bad bprev

File /sys/fs/ufs/ufs_alloc.c

Routine realloccg

Problem The block being reallocated is physical block zero, the boot block.

When reallocating a fragment to a larger size, the routine detects it had been passed a physical block number equal to zero. Physical block number zero is the boot block and cannot be reallocated.

Output Indicates the device, the block size, the physical block number, and the file system. The format is:

```
dev = <0Xd> bsize = <d> bprev = <d> fs = <"string">
```

realloccg: bad optim

File /sys/fs/ufs/ufs_alloc.c

Routine realloccg

Problem The optimization preference of the file is invalid.

This routine extends allocated fragments to a larger size. Before extending the fragment, the routine checks the optimization preference of the file. The valid preferences are for optimized space or optimized time. In this case, the routine detected the value of the optimization preference was neither space nor time and was therefore invalid.

Output Identifies the device, the optimization preference, and the file system. The format is:

```
dev =<0Xd>  optim = <d> fs = <"string">
```

realloccg: bad size

File /sys/fs/ufs/ufs_alloc.c

Routine realloccg

Problem A fragment being reallocated is the wrong size.

When it receives a fragment size, the routine checks it before reallocating the fragment to a larger size. In this case, the routine detected the old or new fragment size was greater than the file system block size or not a multiple of the file system fragment size.

Output Indicates the device, the block size, the old size, the new size requested, and the file system. The format is:

```
dev= <0Xd> bsize= <d> osize= <d> nsize= <d> fs= <"string">
```

receive 1

File /sys/sys/uipc_socket.c

Routine soreceive()

Problem The routine received a null list.

The routine was called, but there was nothing for the routine to do and no memory buffer (mbuf) to process.

receive 1a

File /sys/sys/uipc_socket.c

Routine soreceive

Problem The input memory buffer name for a socket accept is invalid.

This routine receives input memory buffers for use with sockets. In this case, the routine received an input memory buffer name that was invalid because it was not a socket buffer or because the routine could not obtain a valid buffer for it.

receive 2a

File /sys/sys/uipc_socket.c

Routine soreceive

Problem A memory buffer structure does not contain access rights data.

While operating in raw or datagram protocol mode, the routine determines there are two memory buffer (mbuf) structures holding data. However, the routine detected the second mbuf structure did not contain access rights data.

receive 3

File /sys/sys/uipc_socket.c

Routine soreceive

Problem A memory buffer structure does not contain access rights data.

While operating in raw or datagram mode, the routine determines there are three memory buffer (mbuf) structures holding data. However, the routine detected the pointer from the second mbuf structure to the third mbuf structure was zero, indicating the third mbuf structure did not contain access rights data and was therefore invalid.

release_tlbpid: no pid

File /sys/machine/mips/swtch.c

Routine release_tlbpid

Problem The translation lookaside buffer (tlb) identifier is invalid.

While attempting to invalidate or release the tlb identifier associated with this process, the routine detected that the identifier was already invalid.

release_tlbpid: not inuse

File /sys/machine/mips/swtch.c

Routine release_tlbpid

Problem The translation lookaside buffer (tlb) identifier is marked as ''not in use.''

While attempting to invalidate or release the tlb identifier associated with this process, the routine cross-checks to ensure that the identifier is marked as ''in use'' (presumably by this process). During this check, the routine detected that the tlb identifier was marked as ''not in use.''

release_tlbpid: not owner

File /sys/machine/mips/swtch.c

Routine release_tlbpid

Problem The translation lookaside buffer (tlb) identifier is owned by another process.

While attempting to invalidate or release the tlb identifier associated with a process, the routine detected that the tlb identifier was marked as owned by another process.

_Remrq

File /sys/machine/vax/locore.s

Routine remrq

Problem A process is being removed from an empty run queue.

This routine removes processes from the run queue. In this case, the routine detected it was attempting to remove a process from the run queue, but the run queue was empty.

rename: linked directory

File /sys/fs/ufs/ufs_syscalls.c

Routine ufs_rename

Problem A directory being renamed is not empty.

Before a directory can be renamed to another subdirectory of the same name, the destination directory must be empty. The routine first checks whether the link count is greater than two. If it is, the destination directory is not empty. The routine passes the diagnostic `ENOTEMPTY` to the calling routine. Subsequently, the routine made another check of the link count and detected it was still greater than two.

rename: lost dir entry

File /sys/fs/ufs/ufs_syscalls.c

Routine ufs_rename

Problem The source directory for a directory being renamed is lost.

After successfully renaming a directory, the routine removes the links to the source directory. In this case, the routine had entered the new name for the directory and set up the appropriate links, but detected the source directory was lost before it could complete the operation.

resolvfh: cannot resolve file handle

File /sys/fs/nfs/nfs_vfsops.c

Routine nfs_resolvefh

Problem Client did not receive a valid root device file handle.

The client made a request to the mount deamon (mountd) of the server, and the server failed to return a valid file handle.

rksize: invalid partition table

File /sys/io/uba/rk.c

Routine rksize

Problem The partition table of the disk is invalid.

Before determining the size (in blocks) of a partition, the routine checks the partition table. While checking the partition table, the routine detected it was invalid.

rlsize: invalid partition table

File /sys/io/uba/rl.c

Routine rlsize

Problem The partition table of the disk is invalid.

Before determining the size (in blocks) of a partition, the routine checks the partition table. While checking the partition table, the routine detected it was invalid.

rkstrategy: invalid partition table

File /sys/io/uba/rk.c

Routine rkstrategy

Problem The partition table of the disk is invalid.

Before it queues a disk read or write request, the routine first checks the partition table, then retrieves a block number. While checking the partition table, the routine detected it was invalid.

rlstrategy: invalid partition table

File /sys/io/uba/rl.c

Routine rlstrategy

Problem The partition table of the disk is invalid.

Before it queues a disk read or write request, the routine first checks the partition table and then retrieves a block number. While checking the partition table, the routine detected it was invalid.

rmalloc

File /sys/sys/subr_rmap.c

Routine rmalloc

Problem The requested size is invalid.

This routine allocates a variety of kernel resources. While doing so, the routine detected that the resource request has a zero size or, if the resource is SWAPMAP, the size is greater than the configured granularity (dmmax).

rmalloc swapmap

File /sys/sys/subr_rmap.c

Routine rmalloc

Problem A swap area resource address being allocated is not a multiple of CLSIZE.

rmget

File /sys/sys/subr_rmap.c

Routine rmget

Problem A memory resource request has a zero size.

The routine is passed size and resource map parameters that it uses to obtain some resource. The routine detected the size parameter passed to it had a zero size.

rtfree

File	/sys/net/net/route.c
Routine	rtfree
Problem	The pointer to a network routing structure is invalid.
	While attempting to free a network routing structure, the routine discovered the pointer to the structure being freed was zero.

rwgp

File	/sys/fs/gfs/gfs_gnodeops.c
Routine	rwgp
Problem	The command for a gnode operation is invalid.
	This routine reads and writes gnodes. When the routine is passed a command to read and write a gnode, the routine determines whether the command is valid. In this case, the routine detected the command was invalid.

rwsp

File	/sys/fs/specfs/spec_vnodeops.c
Routine	spec_rwgp
Problem	The read/write routine was called without specifying the type of request.
	This routine controls read and write requests for special files. Several arguments are passed to it; one of these is the type of request. The request parameter was not equal to read or write.

rwvp: zero size

File	/sys/fs/nfs/nfs_vnodeops.c
Routine	rwvp
Problem	A block being moved has an invalid size.
	This routine checks the size of blocks being moved. In this case, the routine detected the block being moved had an invalid size of zero.

rzsize: invalid partition table

File /sys/io/scsi/scsi_disk.c

Routine rzsize

Problem The partition table for the disk is invalid.

This routine determines the partition size of a disk (in blocks). Before determining the size of a partition, the routine checks the partition table of the disk. While checking the partition table, the routine detected it was invalid.

rzstrategy: invalid partition table

File /sys/io/scsi/scsi_disk.c

Routine rzstrategy

Problem The partition table for the disk is invalid.

This routine queues read and write request for disks. Before it queues a disk read or write request, the routine checks the partition table of a disk and retrieves a block number. While checking the partition table, the routine detected it was invalid.

saccept

File /sys/sys/uipc_syssocket.c

Routine saccept

Problem A socket connect queue is empty when sockets should be connected to it.

The socket variable, `so_qlen`, indicated there were socket connects on the socket connect queue `so_q`, but the routine detected the socket connect queue was empty.

sbappendrights

File /sys/sys/uipc_socket2.c

Routine sbappendrights

Problem A memory buffer is not available for the operation.

This routine appends additional data (access rights) to a socket buffer. Before appending the access rights, the routine receives a pointer to a memory buffer to use for the operation. The routine detected the value of the pointer returned was zero, indicating there was no memory buffer available for the operation or the value of the access rights was zero.

sbdrop

File /sys/sys/uipc_socket2.c

Routine sbdrop

Problem There are characters in a memory buffer, but no pointer to this data.

This routine drops memory buffers from a socket buffer chain. Before doing so, the routine checks the socket buffer character count and the memory buffer pointer to memory buffer data. The routine detected there was a socket buffer character count but there was no memory buffer pointer to this data.

sbflush

File /sys/sys/uipc_socket2.c

Routine sbflush

Problem A memory buffer being freed is locked for a receive.

While freeing a memory buffer in a socket buffer chain, the routine detected the buffer was locked for a receive.

sbflush 2

File /sys/sys/uipc_socket2.c

Routine sbflush

Problem Freed memory buffers still contain characters.

This routine frees memory buffers in a socket buffer chain. Then, the routine checks for character and memory buffer counts in the socket buffer variables. Next, it checks links in the memory buffer chain pointer. The routine detected at least one of these variables was not zero.

sbi0alert

File sys/machine/vax/locore.s

Routine sbi0alert

Problem The processor detects an SBI0 alert.

When a processor detects an SBI0 alert, it issues an exception and dispatches the error to this routine. In this case, the routine serviced the error by logging it and producing this panic.

sbi0fail

File /sys/machine/vax/locore.s

Routine sbi0fail

Problem A processor detects an SBI0 failure.

When a processor detects an SBI0 failure, it issues an exception and dispatches the failure to this routine. In this case, the routine serviced the failure by logging it and producing this panic.

sbi0flt

File /sys/machine/vax/locore.s

Routine sbi0flt

Problem A processor detects an SBI0 fault.

When a processor detects an SBI0 fault, it issues an exception and dispatches the fault to this routine. In this case, the routine serviced the fault by logging it and producing this panic.

sbi1alert

File /sys/machine/vax/locore.s

Routine sbi1alert

Problem A VAX 8600 detects an SBI1 alert.

When a VAX 8600 detects an SBI1 alert, it issues an exception and dispatches the alert to this routine. In this case, the routine serviced the alert by logging it and producing this panic.

sbi1error

File /sys/machine/vax/locore.s

Routine sbi1error

Problem A VAX 8600 detects an SBI1 error.

When a VAX 8600 detects an SBI1 error, it issues an exception and dispatches the error to this routine. In this case, the routine serviced the error by logging it and producing this panic.

sbi1fail

File /sys/machine/vax/locore.s

Routine sbi1fail

Problem A VAX 8600 detects an SBI1 failure.

When the VAX 8600 detects an SBI1 failure, it issues an exception and dispatches the failure to this routine. In this case the routine serviced the failure by logging it and producing this panic.

sbi1flt

File /sys/machine/vax/locore.s

Routine sbi1flt

Problem A VAX 8600 detects an SBI1 fault.

When a VAX 8600 detects an SBI1 fault, it issues an exception and dispatches the fault to this routine. In this case, the routine serviced the fault by logging it and producing this panic.

sbia0error

File /sys/machine/vax/locore.s

Routine wtime

Problem A VAX 8600 detects an SBIA0 error.

When the VAX 8600 detects an SBIA0 error, it issues an exception and dispatches the error to this routine. In this case, the routine serviced the exception by producing this panic.

secondary cpu requested

File /sys/machine/vax/machdep.c

Routine cpu_ip_intr

Problem A non-boot processor requested the system to be panicked.

secondary cpu requested

File /sys/sys/kern_cpu.c

Routine start_one_cpu

Problem A non-boot processor requested the boot processor to panic the system.

scs - bad connid seen during connection abortion

File /sys/io/scs/scs_protocol.c

Routine scs_abort_conn

Problem The connection block identification number is invalid.

scs - broken sanity timer

File See Table 2-10

Routine See Table 2-10

Problem The Systems Communications Subsystem (SCS) sanity timer is either already disabled, or it was not previously enabled.

The routines that can issue this panic are briefly described in Table 2-10.

Table 2-10: Sanity Timer Checks

File	Routine	Description
/sys/io/scs/scs_protocol.c	scs_receive	Receives SCS sequenced messages
	scs_request	Transmits SCS requests
	scs_timer	Oversees SCS timer-related functions

scs - corrupted listening sysap queue

Files /sys/io/scs/scs_event.c, /sys/io/scs/scs_protocol.c

Routines scs_new_path, scs_receive

Problem The listening connection is in an invalid state.

Two routines can issue this panic. The `scs_new_path` routine processes new path notifications and the `scs_receive` routine receives System Communications Subsystem (SCS) sequenced messages.

While performing their respective tasks, one of the routines determined that the listening connection is in an invalid state.

scs - corrupted sca configuration database

File /sys/io/scs/scs_event.c

Routine scs_path_crash

Problem Unable to retrieve the connection block.

This routine processes path failure notifications. While cleaning up a connection, the routine discovers it is unable to retrieve the connection block.

scs - invalid asynchronous event on connection

File /sys/io/scs/scs_subr.c

Routine scs_init_cmsb

Problem The SYSAP event is invalid.

This routine prepares interface data structures for asynchronous event SYSAP notifications. While preparing for an event notification, the routine determined that the event is invalid.

scs - invalid connection state

File /sys/io/scs/scs_conn.c, /sys/io/scs/scs_event.c

Routine scs_disconnect, scs_path_crash

Problem The System Communication Subsystem (SCS) connection is illegal.

Two routines can issue this panic. The `scs_disconnect` routine terminates SCS connections and the `scs_path_crash` routine processes path failure notifications.

While performing its tasks, one of these routines determined that the SCS connection is illegal.

scs - unexpected connection abortion occurred

File /sys/io/scs/scs_protocol.c

Routine scs_abort_conn

Problem The Systems Communication Subsystem (SCS) connection state is invalid.

This routine completes SCS connection establishment abortions. While completing the termination of a connection, the routine determined the following:

- The connection state is inappropriate.
- The connection should not be aborted at this time.

scs - unknown console logging formatting code

File /sys/io/scs/scs_error.c

Routine scs_console_log

Problem This routine optionally logs SCS events to the console terminal. While logging an event, the routine determined that the class of variable information to be logged is unknown.

scs - unknown/invalid event code

File /sys/io/scs/scs_error.c

Routine scs_console_log

Problem This routine optionally logs SCS events to the console terminal. While logging an event, the routine determined one of the following:

- The event type is unknown.
- The event is not an SCS-specific event.
- The event severity level is invalid.
- The event is unknown to SCS.
- The event is not supposed to be logged by SCS.

scs - unknown scs message type requested

File /sys/io/scs/scs_protocol.c

Routine scs_request, scs_response

Problem The System Comunications Subsystem (SCS) response type is unknown.

Two routines can issue this panic. The `scs_request` routine transmits SCS requests and the `scs_response` routine transmits SCS responses.

While performing its function, one of these routines determined that the response type is unknown.

scs$directory - accept failed

File /sys/io/sysap/scs_directory.c

Routine scs$dir_control

Problem The SCS$DIRECTORY connection request failed.

This routine processes SCS$DIRECTORY event notifications. The panic is issued when the routine determines that the acceptance failed for an unsatisfactory reason.

scs$directory - disconnect failed

File /sys/io/sysap/scs_directory.c

Routine scs$dir_control and scs$dir_receive

Problem The SCS$DIRECTORY disconnect failed for an unacceptable reason.

Two routines can issue this panic. The `scs$dir_control` routine processes SCS$DIRECTORY event notifications and the `scs$dir_receive` routine receives SCS$DIRECTORY messages.

When either of the routines fails to successfully complete its funtion, it issues this panic.

scs$directory - reject failed

File /sys/io/sysap/scs_directory.c

Routine scs$dir_control

Problem The SCS$DIRECTORY connection reject failed.

This routine processes SCS$DIRECTORY event notifications. While rejecting a connection request, the routine determined that the rejection failed.

scs$directory - response transmission failure

File /sys/io/sysap/scs_directory.c

Routine scs$dir_receive

Problem The SCS$DIRECTORY messages transmission failed.

This routine receives SCS$DIRECTORY messages. While transmitting a message, the routine determined that the transmission failed for an unacceptable reason.

scs$directory - unknown event

File /sys/io/sysap/scs_directory.c

Routine scs$dir_control

Problem The SCS$DIRECTORY event is unknown.

This routine processes SCS$DIRECTORY event notifications. While processing such an event, the routine determined that the event is unknown.

scsnet - ACCEPT Failed

File /usr/src/sys/if_scs.c

Routine scsnet_control

Problem The SCS network driver return status failed.

The SCS network driver attempted to accept a connection request from a remote system, but the return status was a failure. It is likely the connection was in an invalid state.

scsnet - Can't Find Recv System in Database

File /usr/src/sys/if_scs.c

Routine scsnet_control

Problem The SCS network driver attempted to accept a connection request from a remote system.

Before it could do so, the driver tried to obtain information about the remote system from the local SCS database. There was no information in the local database, which implies that the local system has not heard from the remote system at all.

scs_net - conn recv, too many systems

File /usr/src/sys/if_scs.c

Routine scsnet_control

Problem Unable to allocate resources for the connection.

A connect attempt from a remote system was accepted, but the driver was unable to allocate local resources for the connection. There should be enough available resources for the number of nodes specified by the parameter SCSNET_MAXHOSTS.

scsnet - control

File /usr/src/sys/if_scs.c

Routine scsnet_control

Problem The SCS network driver attempted to accept a connection request from a remote system.

Before it could do so, the driver tried to obtain information about the remote system from the local SCS database. There was no information in the local database, which implies that the local system has not heard from the remote system at all.

scs_net: - no space for new system

File /usr/src/sys/if_scs.c

Routine scsnet_control

Problem Unable to allocate resources for the new connection.

The SCS network driver was notified of a new path to a remote system but was unable to allocate resources for the new connection. There should be enough resources for all possible connections through a star coupler.

scsnet - reject

File /usr/src/sys/if_scs.c

Routine scsnet_control

Problem The SCS network driver attempted to reject a connection request from a remote system.

The reject was issued to the SCS subsystem, but the return status was failure.

scsnet - REJECT Failed

File /usr/src/sys/if_scs.c

Routine scsnet_control

Problem The SCS network driver attempted to reject a connection request from a remote system.

The reject was issued to the SCS subsystem, but the return status was failure. This failure typically indicates that the connection was in an invalid state for the reject to succeed.

scs_net - SCS Disconnect Failed

File /usr/src/sys/if_scs.c

Routine scsnet_control

Problem Path to the remote system failed.

The connection to the remote system was terminated for one of the following reasons: the remote system requested that the connection be terminated; a connection was previously established to the remote system; or, local memory resources were not available to establish a connection.

scsnet: block transfer dup

File /usr/src/sys/if_scs.c

Routine scsnet_msgevent

Problem The local scsnet driver was notified for the second time by the remote system that a block transfer has completed.

scsnet: NO ROOM in tail mbuf

File /usr/src/sys/if_scs.c

Routine scsnet_output

Problem Overflowed the local driver buffer.

This panic occurred while copying an output packet to a local driver buffer that was not large enough for the packet.

scsnet: proto header too long

File /usr/src/sys/if_scs.c

Routine scsnet_output

Problem The transport level protocol header was larger than that expected by the scsnet driver output routine.

scsnet_event - unknown proto

File /usr/src/sys/if_scs.c

Routine scsnet_control, scsnet_dgevent

Problem The local driver received a block or datagram from the remote system.

The protocol type field of transfer contained an invalid protocol type. Only Internet protocols are supported.

sdc: zero pfn in pte

File /sys/io/uba/sdc.c

Routine sdustart

Problem A page table contains invalid entries.

The routine maps memory from page table entries. The delimiter for the page table is a page table entry containing zero in its page frame bits. While searching the page table, the routine detected a page table entry containing zero in its page frame bits before the delimiter for the page table.

sdsize: invalid partition table

File /sys/io/uba/sdc.c

Routine sdsize

Problem The partition table of the disk is invalid.

Before determining the size (in blocks) of a partition, the routine checks the partition table. While checking the partition table, the routine detected it was invalid.

sdstragegy: invalid partition table

File /sys/io/uba/sdc.c

Routine sdstrategy

Problem The partition table of the disk is invalid.

Before it queues a disk read or write request, the routine first checks the partition table and then retrieves a block number. While checking the partition table, the routine detected it was invalid.

setblock

File /sys/fs/ufs/ufs_subr.c

Routine setblock

Problem A free block has an invalid number of fragments.

This routine adds a fragment to the free block map for a cylinder. When it finds a free block, the routine checks the block for the number of fragments it contains. Then, the routine compares that number to the number of fragments allowed by the file system. In this case, the routine detected the free block had an invalid number of fragments. The number of fragments per block can be only 8, 4, 2, or 1.

setdlim

File /sys/fs/gfs/gfs_sysquota.c

Routine setdlim

Problem The uid values in a disk quota and its disk quota structure do not match.

When a disk quota is allocated and linked to its disk quota structure, the uid values in both structures are set to the same value to show they have the same owner. While setting disk quota limits, the routine detected these uid values did not match.

setduse

File /sys/fs/gfs/gfs_sysquota.c

Routine setduse

Problem The uid values in a disk quota and its disk quota structure do not match.

When a disk quota is allocated and linked to its disk quota structure, the uid values in both structures are set to the same value to show they have the same owner. While setting disk usage limits, the routine detected these uid values did not match.

setrq

File /sys/machine/vax/locore.s

Routine _Setrq

Problem A process on the run queue is not in the run state.

This routine puts processes on the run queue. While doing so, the routine checks the state of the processes. In this case, the routine detected the process it was putting on the run queue was not in the run state.

setrq p_rlink

File /sys/machine/mips/swtch.c

Routine setrq

Problem A process is already linked to run the queue.

The system attempted to place a process on the run queue that was already linked to the run queue.

setrun

File /sys/sys/kern_synch.c

Routine setrun

Problem A process being started is in an invalid state.

This routine starts a process running. While doing so, the routine detected the process was in an invalid state because it was not stopped, sleeping, or idle.

setwarn

File	/sys/fs/gfs/gfs_sysquota.c
Routine	setwarn
Problem	The uid values in a disk quota and its disk quota structure do not match.
	When a disk quota is allocated and linked to its disk quota structure, the uid values of both structures are set to the same value to show they have the same owner. While setting disk quota warning levels, the routine detected these uid values did not match.

sleep

File	/sys/sys/kern_synch.c
Routine	sleep
Problem	A user process cannot be put to sleep.
	This routine puts user processes to sleep. In this case, the routine was called to put a process to sleep, but could not do so.

smat: alloc p_sm

File	/sys/sys/uipc_smem.c
Routine	smat
Problem	The kernel memory allocator (km_alloc) failed to allocate memory for the shared memory information of the process (a km_alloc() problem).

smat: smbeg

File	/sys/sys/uipc_smem.c
Routine	smat
Problem	NULL pointer to shared memory information.
	The routine detected a process that indicated it had attached shared memory, but the process has a NULL pointer to shared memory information in the process (proc) structure.

sm_attach: out of segments

File /sys/vm/mips/pt_machdep.c

Routine sm_attach

Problem No per-process shared memory data structures were found for the segment.

While attempting to attach a shared memory segment to the process, the routine did not find any per-process shared memory data structures for segment use. This occurs when the user does not specify an attach address.

sm_attach: out of segments 2

File /sys/vm/mips/sm_machdep.c

Routine sm_attach

Problem No per-process shared memory data structures were found for the segment.

While attempting to attach a shared memory segment to the process, the routine did not find any per-process shared memory data structures for segment use. This occurs when the user does not specify an attach address.

smccdec: rssize

File /sys/vm/vm_smem.c

Routine smccdec

Problem The physical memory of a shared memory segment has been released but its resident set size is not zero.

This routine decrements the usage count for memory-resident shared memory segments. When the count reaches zero, the associated shared memory is released. In this case, the routine released the shared memory segment but detected the resource set size for the segment was not zero.

smccdec: smseg

File /sys/vm/vm_smem.c

Routine smccdec

Problem A shared memory segment is not found in the process structures linked to it.

This routine decrements the usage count for memory-resident shared data segments. When it receives a specific shared memory segment, the routine checks that the segment has a process linked to it. In this case, the routine did not find the segment in any process structures linked to it.

smclean: p_sm

File /sys/vm/vm_smem.c

Routine smclean

Problem NULL pointer to shared memory information.

The routine detected a process that indicated it had attached shared memory, but the process has a NULL pointer to shared memory information in the process (proc) structure.

sm_del_psm: smcount

File /sys/vm/vm_smem.c

Routine sm_del_psm

Problem Negative shared segment count for the attached process.

While detaching a shared memory segment from the virtual address space of a process, the routine detected a negative shared segment count for the attached process.

smdt: p_sm

File /sys/sys/uipc_smem.c

Routine smdt

Problem NULL pointer to shared memory information.

The routine detected a process that indicated it had attached shared memory, but the process has a NULL pointer to shared memory information in the process (proc) structure.

smfork: cpsmp

File /sys/vm/vm_smem.c

Routine smfork

Problem NULL pointer to shared memory information.

A child process, while attempting to copy shared memory information from its parent, has a NULL pointer to shared memory information in the process structure.

smfork: ppsmp

File /sys/vm/vm_smem.c

Routine smfork

Problem NULL pointer to shared memory information.

A child process, while attempting to copy shared memory information from its parent, has a NULL pointer to shared memory information in the process structure.

smfree rssize

File /sys/vm/vm_smem.c

Routine smfree

Problem The physical memory of a shared memory segment has been released, but its resident set size is not zero.

This routine releases shared memory segments. During such a release, the routine detected the resource set size for the segment was not zero after releasing the memory.

sm_ins_psm

File /sys/vm/vm_smem.c

Routine sm_ins_psm

Problem Shared segment count has exceeded the system limit.

While attaching a shared memory segment to the virtual address space of a process, the routine detected that the count of shared segments to which the process is currently attached has exceeded the system-imposed limit.

sm_ins_psm: too many segments per process

File /sys/vm/vm_smem.c

Routine sm_ins_psm

Problem Array overflow in the per-process shared memory data structure.

While attaching a shared memory segment to the virtual address space of a process, the routine detected that an array overflow occurred in the shared memory data structure of the process.

smlink

File	/sys/vm/vm_smem.c
Routine	smlink
Problem	A shared memory segment is not found in the process structures linked to it.
	This routine adds a process to the list of processes already sharing a memory segment. In this case, the routine did not find the segment in the process structure.

smlink: p->p_sm

File	/sys/vm/vm_smem.c
Routine	smunlink
Problem	NULL pointer to shared memory information.
	While attempting to link a process to a shared memory segment, the routine detects that the process has a NULL pointer to shared memory information in the process structure.

smlink: p_sm

File	/sys/vm/vm_smem.c
Routine	smlink
Problem	NULL pointer to shared memory information.
	While attempting to link a process to a shared memory segment, the routine detects that the process has a NULL pointer to shared memory information in the process structure.

smlink: q->p_sm

File	/sys/vm/vm_smem.c
Routine	smunlink
Problem	NULL pointer to shared memory information.
	While attempting to unlink a process from a shared memory segment, the routine detected a process that has a NULL pointer to shared memory information.

smp_lock_long: invalid lock type

File	/sys/sys/kern_lock.c
Routine	check_lock
Problem	The lock structure contains an undefined lock type. This can be caused by an uninitialized lock or one that has been written over.

smp_lock_long: lock owner

File /sys/sys/kern_lock.c

Routine smp_lock_long

Problem A processor already owns the lock it is trying to aqcuire.

smp_lock_long: lock position messup

File /sys/sys/kern_lock.c

Routine check_lock

Problem A processor is attempting to acquire a lock in an incorrect order. Locks must be asserted in decreasing position number.

smp_lock_long: wrong ipl

File /sys/sys/kern_lock.c

Routine check_lock

Problem A processor is attempting to acquire an smp lock at a system priority level that is below the minimum level.

smp_lock_long: beyond max wait count

File /sys/sys/kern_lock.c

Routine smp_lock_long

Problem A processor has timed out waiting to assert a lock. This is usually an indication that another processor has hung holding an smp lock.

smp_unlock_long: not lock owner

File /sys/sys/kern_lock.c

Routine smp_unlock_long

Problem A processor tried to unlock an SMP lock that it didn't have locked. This indicates a locking problem in the kernel.

smp_unlock_long: invalid lock address

File /sys/sys/kern_lock.c

Routine smp_unlock_long

Problem The lock structure contains an undefined lock type. This can be caused by an unintialized lock or one that has been written over.

smp_unlock_long: no process woken

File	/sys/sys/kern_lock.c
Routine	smp_unlock_long
Problem	An SMP sleep lock had a non-zero wanted field, but no process was waiting for the lock.

smp_unlock: lock not held

File	/sys/machine/vax/lock.s
Routine	Smp_unlock
Problem	A processor tried to unlock an SMP lock that it did not have locked. This indicates a locking problem in the kernel.

smp_unlock: no process woken

File	/sys/machine/vax/lock.s
Routine	Smp_unlock
Problem	An SMP sleep lock had a non-zero wanted field, but no process was waiting for the lock.

smunlink #1

File	/sys/vm/vm_smem.c
Routine	smunlink
Problem	A shared memory segment is not found in the process structure linked to it. This routine removes a process from the list of processes sharing a memory segment. In this case, the routine did not find the segment in the process structure.

smunlink #2

File	/sys/vm/vm_smem.c
Routine	smunlink
Problem	A shared memory segment is not found in the process structure linked to it. This routine removes a process from the list of processes sharing a memory segment. In this case, the routine could not find the segment in the process structure linked to it.

sm_retrieve_sa: Could not find SMS in proc

File sys/vm/vm_smem.c

Routine sm_retrieve_sa

Problem A shared memory segment is not found in the process structures linked to it.

sm_retrieve_sa: p_sm == (struct p_sm *) NULL

File sys/vm/vm_smem.c

Routine sm_retrieve_sa

Problem Attempting to retrieve the starting address of a shared memory segment, the routine detects that the process has a NULL pointer to the shared memory segment.

SM_UNLOCK: shared memory not locked

File sys/h/vmmac.h

Routine macro definition SM_UNLOCK

Problem The system has detected that a shared memory segment that it is unlocking is not locked. This violates the locking conventions for shared memory segments.

soaccept: !NOFDREF

File /sys/sys/uipc_socket.c

Routine soaccept

Problem A socket being accepted is already open.

While accepting a socket, the routine checks and then clears the `NOFDREF` socket state bit to ensure there is a file table reference to the open socket. However, when checking the socket state bit, the routine detected it was already clear.

soclose: NOFDREF

File /sys/sys/uipc_socket.c

Routine soclose

Problem A socket being closed is already closed.

While closing a socket, the routine checks and then sets the `NOFDREF` socket state bit to ensure there is no longer a file table reference to the closed socket. However, when checking the socket state bit, the routine detected it was already set.

sofree dq

File /sys/sys/uipc_socket.c

Routine sofree

Problem There is a pointer to a socket entry, but there are no entries in the socket accept queues.

Before freeing a socket, the routine discovers the `so_head` pointer was pointing to an accept socket entry. However, the routine then checked the two socket accept queues for an accept_socket entry and found none.

softclock: invalid affinity

File /sys/sys/kern_clock.c

Routine softclock

Problem A processor had a timeout on its queue that did not belong to it.

soisconnected

File /sys/sys/uipc_socket2.c

Routine soisconnected

Problem A socket being connected is not in the partial connect queue.

This routine moves sockets from the partial connect queue to the connect queue. The routine determines a socket is connected and has a `so_head` pointer to an accept socket entry. While moving the socket to the connect queue, the routine could not find the socket in the partial connect queue.

spec_select

File /sys/fs/specfs/spec_vnodeops.c

Routine spec_select

Problem The generic node (gnode) type was not equal to a character device.

This routine is called by the select system call for all character special devices. However, the routine detected a block device that should have been processed at a higher level.

st0: zero pfn in pte

File /sys/io/uba/sg.c

Routine sg_strategy

Problem A page table contains invalid entries.

The routine maps memory from page table entries. The delimiter for the page table is a page table entry that contains zero in its page frame bits. While searching the page table, the routine detected an entry with zero in its page frame bits before the delimiter for valid page table entries.

st0: zero pfn in pte

File /sys/io/uba/stc.c

Routine stintr

Problem A page table contains invalid entries.

The routine maps memory from page table entries. The delimiter for the page table is a page table entry that contains zero in its page frame bits. While searching the page table, the routine detected an entry with zero in its page frame bits before the delimiter for valid page table entries.

st0: zero pfn in pte

File /sys/io/uba/stc.c

Routine st_start

Problem A page table contains invalid entries.

The routine maps memory from page table entries. The delimiter for the page table is a page table entry that contains zero in its page frame bits. While searching the page table, the routine detected an entry with zero in its page frame bits before the delimiter for valid page table entries.

swalloc_vtod: SMEM

File vm/vm_drum.c

Routine swalloc_vtod

Problem A shared memory segment cannot be found. This routine allocates swap space(disk blocks) for the given virtual page number. While doing so, the routine failed to find a shared memory segment for a specified shared memory virtual address.

swalloc_vtod: Can not classify page

File vm/vm_drum.c

Routine swalloc_vtod

Problem The user virtual page cannot be classified as text, data, stack, or shared memory. While attempting to convert a virtual page number to disk block number, the routine could not classify the page as text, data, stack, or shared memory

swap bad pte

File /sys/vm/vm_swp.c

Routine swap

Problem Modified (dirty) pages cannot be written because a page table entry is invalid.

This routine determines there are modified (dirty) pages that must be written to disk (paged out). After doing so, the routine detected the page table entry contained invalid data, preventing the modified pages from being swapped.

swapconf: Cannot open swap device

File /sys/machine/mips/swtch.c

Routine swapconf

Problem While configuring the swap devices, the kernel could not access a swap device.

swapconf: km_alloc swapmap

File /sys/sys/init_main.c

Routine swapconf

Problem The kernel memory allocator failed to allocate memory for the swapmap (a km_alloc() problem).

swapin

File /sys/vm/vm_swap.c

Routine swapin

Problem The user area (uarea) page table entries (pte) are corrupted.

This routine swaps a process into main memory. After swapping the pte supporting the user area, the routine has detected that the uarea page table entries have been corrupted.

swapin: p_sm#

File /sys/vm/vm_swap.c

Routine swapin

Problem NULL Pointer to shared memory information.

The routine detected a process that indicated it had attached shared memory, but the process has a NULL pointer to shared memory information in the process (proc) structure.

swapout

File /sys/vm/vm_swap.c

Routine swapout

Problem A process being swapped out is not marked as swapped out and is either not runable or not currently running.

This routine swaps out a process from main memory to disk. After swapping out the process, the routine checks the process state. During this check, the routine found the state inconsistent.

swapout: p_sm

File /sys/vm/vm_swap.c

Routine swapout

Problem NULL Pointer to shared memory information.

The routine detected a process that indicated it had attached shared memory, but the process has a NULL pointer to shared memory information in the process (proc) structure.

swapout rssize

File /sys/vm/vm_swap.c

Routine swapout

Problem A process is swapped out, but its resident set size is not equal to zero.

This routine swaps out a process from main memory to disk. After doing so, the routine checks the resident set size for the process. In this case, the routine detected the resident set size for the process was not equal to zero.

swdspt

File /sys/vm/vm_swap.c

Routine swdspt

Problem A page table entry (pte) is corrupted.

While swapping the user page tables, the routine detected that a pte was corrupted.

swdspt: data

File /sys/vm/vm_swap.c

Routine swdspt

Problem The page table entries are not clear.

This routine swaps out data and stack page tables from main memory to disk and clears the page table entries that map those page tables. During this function, the routine detected that the page table entries are not clear for the data.

swdspt: stack

File /sys/vm/vm_swap.c

Routine swdspt

Problem The page table entries are not clear.

This routine swaps out data and stack page tables from main memory to disk and clears the page table entries that map those page tables. During this function, the routine detected that the page table entries are not clear for the stack.

swfree

File /sys/vm/vm_sw.c

Routine swfree

Problem The number of blocks being freed is greater than the number of swap blocks.

This routine frees blocks from the swap map. While doing so, the routine detected that the number of blocks to be freed on the swap map was greater than the total number of swap blocks.

swfree: no swapmap

File /sys/vm/vm_sw.c

Routine swfree

Problem No swapmap allocated.

When the routine attempts to initialize a swapmap, it detected that no swapmap was allocated

swtch doesn't hold lk_rq

File /sys/sys/kern_lock.c

Routine swtch_check

Problem A processor was going to reschedule without holding the run queue locked.

swtch holds spin lock

File /sys/sys/kern_lock.c

Routine swtch_check

Problem A processor attempted to reschedule holding a spin lock. This is not allowed because it could cause the system to deadlock.

swstrategy

File /sys/vm/vm_sw.c

Routine swstrategy

Problem A page kluster is being swapped out to an invalid device.

This routine locates the swap device strategy routine to use for swapping paging. The routine calculated the device number, but the result was zero. The device was invalid.

swtch

File /sys/machine/vax/locore.s

Routine _Swtch

Problem A process run queue or process argument is invalid.

This routine saves context for processes as they are switched on and off the run queue. While switching a process, the routine detected an element of the run queue or a process argument was invalid.

sys pt too small

File /sys/machine/vax/machdep.c

Routine startup

Problem The system page table is too small for the configured physical memory.

The routine detects the system page table is too small. Then, the routine reduces the size of physical memory to the minimum required for configuration. After memory has been reconfigured, the routine detected the system page table was still too small.

Output Identifies the problem and shows the amount of physical memory now available. The format is:

```
System page table too small, reducing memory to <%d> meg
```

syscall

File /sys/machine/vax/trap.c

Routine syscall

Problem A system call is issued from kernel mode.

This routine checks all system calls and detects whether they are issued from user mode or kernel mode. In this case, the routine detected the system call was made from kernel mode. System calls must be issued from user mode.

szstart: v

File /sys/b.mips/mips/scsi.c

Routine szstart()

Problem Data transfer value is zero.

When the routine checks the user area address for data input or output, the beginning transfer value is zero. Because the data value passed back by the virtual memory code is a fixed constant of zero, this panic should never occur.

sz_start: zero pfn in pte

File /sys/io/scsi/scsi.c

Routine sz_start

Problem An SCSI driver page table contains invalid entries.

The routine maps memory from page table entries in order to copy data to or from the buffers of the users. The delimiter for the page table is a page table entry that contains zero in its page frame bits. While searching the page table, the routine detected an entry with zero in its page frame bits, before the delimiter for valid page table entries.

tcp_closekeepinp not lock owner

File tcp_subr.c

Routine tcp_closekeepinp

Problem The socket referenced by a tcp control block is not locked.

tcp_output

File /sys/net/netinet/tcp_output.c

Routine tcp_output

Problem There is a NULL pointer to an output buffer containing data.

This routine sends data packets. In this case, the routine had data to send, but detected a NULL pointer to the output buffer containing the data.

tcp_output REXMT

File /sys/net/netinet/tcp_output.c

Routine tcp_setpersist

Problem A retransmit timer is set when it should be cleared.

Before the routine is entered, the retransmit (REXMT) timer in the `tcp` control block structure is cleared. Then, the routine checks the REXMT timer before resetting it. In this case, the routine detected the REXMT timer was still set.

tcp_pulloobxti no m0

File tcp_input.c

Routine tcp_pulloobxti

Problem The pointer to out-of-band data is NULL.

tcp_pulloobxti no m->m_next

File tcp_input.c

Routine tcp_pulloobxti

Problem The pointer to out-of-band data is NULL.

tcp_pulloutofband

File /sys/net/netinet/tcp_input.c

Routine tcp_pulloutofband

Problem There is a request to process urgent data but no urgent data is present.

This routine handles requests to process urgent data. The routine receives the request and checks the `urgent_count` variable in the `tcpiphdr` structure. In this case, the routine found no urgent data was present in the structure.

Text Corruption gp != x_gptr

File /sys/vm/vm_text.c

Routine xalloc

Problem A text pointer and its associated gnode pointer do not match the gnode pointer passed in.

This routine adds processes to a list of processes sharing a text segment. As it was adding a process to the list, the routine detected the text pointer and the associated gnode pointer did not match the gnode pointer that it was passed.

text rssize

File /sys/vm/vm_text.c

Routine xccdec

Problem The physical memory of a shared text segment has been released, but its resident set size is not zero.

This routine decrements the usage count for memory-resident shared text segments. When the count reaches zero, the routine releases the physical memory of the associated shared text segment. In this case, the routine released the physical memory but detected the resident set size for the segment was not zero.

timeout table overflow

File /sys/sys/kern_clock.c

Routine timout

Problem A timeout table overflow prevents a function from being rescheduled.

This routine schedules a function call at a specified time. While it was loading the timeout table, the routine detected the timeout table overflowed. The overflow prevented the function call from being rescheduled.

tlbmiss no tlbpid assigned

File /sys/machine/mips/trap.c

Routine tlbmiss

Problem The translation lookaside buffer (tlb) identifier has not been assigned.

While servicing a virtual address in user space, the routine detected that a tlb identifier has not been assigned.

tlbmiss on invalid kernel page

File /sys/machine/mips/trap.c

Routine tlbmiss

Problem While servicing a virtual address in KSEG2 space (mapped system address space), the routine detected that the associated page table entry was invalid.

tlbmiss page table not valid

File /sys/machine/mips/trap.c

Routine tlbmiss

Problem Invalid page table entry (pte).

While servicing a virtual address in KPTESEG space, the routine detected that the associated pte was invalid.

tlbmod on invalid pte

File /sys/machine/mips/trap.c

Routine tlbmod

Problem Invalid page table entry (pte).

While attempting to set the dirty flag in the pte, the routine detected that the associated pte was invalid.

Too many EBOX errors to recover...

File	/sys/machine/vax/ka8600.c
Routine	eboxserv
Problem	The VAX 8600 issues three EBOX errors within 10 milliseconds. This routine counts the number and proximity of EBOX errors from the VAX 8600 processor. EBOX errors are recoverable unless three of them occur within 10 milliseconds. In this case, the routine detected three EBOX errors within 10 milliseconds.
Output	Identifies the machine check type code and other diagnostic information. See the *Guide to the Error Logger System* for more information.

Too many generic machine checks to recover

File	/sys/machine/vax/ka8600.c
Routine	genericserv
Problem	The VAX 8600 issues two MBOX 1D errors within 10 milliseconds. This routine counts the number and proximity of MBOX 1D errors issued by the VAX 8600 processor. MBOX 1D errors are recoverable unless two occur within 10 milliseconds. In this case, the routine detected two MBOX 1D errors within 10 milliseconds.
Output	Identifies the machine check type code and other diagnostic information. See the *Guide to the Error Logger System* for more information.

Too many IBOX errors to recover...

File	/sys/machine/vax/ka8600.c
Routine	iboxserv
Problem	The VAX 8600 issues three IBOX errors within 10 milliseconds. This routine counts the number and proximity of IBOX errors issued by the VAX 8600 processor. IBOX errors are recoverable unless three of them are issued within 10 milliseconds. In this case, the routine detected three IBOX errors within 10 milliseconds.
Output	Identifies the machine check type code and other diagnostic information. See the *Guide to the Error Logger System* for more information.

Too many machine check errors to recover...

File /sys/machine/vax/ka8600.c

Routine ka8600machcheck

Problem More than two VAX 8600 machine checks.

This routine counts machine checks issued by the VAX 8600 processor. The routine detected more than two machine checks had been issued by the KA-8600 processor.

Output Identifies the machine check type codes and other diagnostic information. See the *Guide to the Error Logger System* for more information.

Too many MBOX errors to recover...

File /sys/machine/vax/ka8600.c

Routine mboxserv

Problem An MBOX error results in a fatal VAX 8600 machine check.

This routine detects machine checks issued by the VAX 8600 processor. When the routine detects an MBOX error, no recovery is possible.

Output Identifies the machine check type code and other diagnostic information. See the *Guide to the Error Logger System* for more information.

too many systems

File /usr/src/sys/if_scs.c

Routine scsnet_init

Problem Too many systems.

During initialization, the scsnet driver discovered more systems than it was prepared to handle.

Too many BI errors

File /sys/io/bi/biinit.c

Routine bierrors

Problem The number of BI errors exceeds 65536.

The routine checks the number of BI errors during the boot process. While checking the number of BI errors, the routine detected they number more than 65536. BI errors are not counted after the boot process is complete.

trap

File	/sys/machine/vax/trap.c
Routine	trap
Problem	A processor-detected trap is either not recoverable or an unknown type. This routine handles processor-detected traps and determines whether they are recoverable. There are several trap types that are not recoverable. See Table 2-11 for a list of each trap type and the panic string printed when the routine handles these traps. A type constant indicates the trap type detected by the processor. For arithmetic traps and compatibility mode faults (trap types 6 and 11, respectively), trap type codes are also significant. The routine handles an unknown trap type code by calling `panic` and passing `trap` as the argument. All other traps that cause a call to panic include the panic string shown in Table 2-11 for the trap detected.
Output	Identifies the trap type, the trap type code, and the program counter address. The format is: `trap type <d>, code = <0Xd>, pc = <0Xd>` Other output is generated through the error logging facility. See the *Guide to the Error Logger System* for more information.

Table 2-11: Trap Types and Panics

Type	Trap Panic String (and Type Code)
0	Reserved addressing mode
1	Privileged instruction
2	Reserved operand
3	Breakpoint
4	Xfc trap
5	Syscall trap
6	Arithmetic fault 1 = integer overflow trap 2 = integer divide-by-zero trap 3 = floating point overflow trap 4 = floating point/decimal divide by zero trap 5 = floating point underflow trap 6 = decimal overflow trap 7 = subscript range trap 8 = floating point overflow fault 9 = floating point divide-by-zero fault A = floating point underflow fault
7	AST trap
8	Segmentation fault
9	Protection fault
10	Trace trap

Table 2-11: (continued)

Type	Trap Panic String (and Type Code)
11	Compatibility mode trap 0 = reserved instruction 1 = BPT instruction 2 = IOT instruction 3 = EMT instruction 4 = TRAP instruction 5 = invalid instruction 6 = odd address abort
12	Page fault
13	Page table fault

trap

File /sys/machine/mips/trap.c

Routine trap()

Problem The sytem received a trap exception without a specific error bit set in the cause register.

tsintr

File /sys/io/uba/ts.c

Routine tsintr

Problem An interrupt operation is invalid.

After completing an interrupt operation, the routine first checks the type of interrupt operation and then updates the block number to show what operation caused the interrupt. While checking the type of interrupt operation, the routine detected it was unknown.

ttrstrt

File /sys/sys/tty.c

Routine ttrstrt

Problem A tty structure is needed, but cannot be found.

The routine receives a pointer to a tty structure from the `timeout` routine. The routine detected the pointer was zero, indicating there was no tty structure.

ttwrite

File /sys/sys/tty.c

Routine ttwrite

Problem There is data to write to a terminal, but no I/O vectors contain data.

The I/O count in the I/O structure indicated there were no vectors holding data, but the I/O structure of the user indicated there was more data to be written.

ttyrub

File /sys/sys/tty.c

Routine ttyrub

Problem An input character being deleted cannot be deleted.

This routine deletes (rubs out) input characters. To do so, the routine checks the `partab` data structure to determine the operation to be performed on the character. The routine detected the value of the character did not match one of the cases defined in the partab structure.

uba crazy

File /sys/io/uba/uba.c

Routine ubaerror

Problem The UNIBUS adapter has been reset 500 times.

Output The routine counts the number of times a UNIBUS adapter is reset since the system was rebooted. While checking the count, the routine detected it has reached 500. Reboot the system to reinitialize the counter.

uba zero uentry

File /sys/io/netif/if_qe.c

Routine qbsetup

Problem A page table contains invalid entries.

The routine sets up the Q-Bus map registers from page table entries. The delimiter for the page table is a page table entry containing zero in its page frame bits. While searching the page table, the routine detected a page table entry containing zero in its page frame bits before the delimiter for valid page table entries.

uba zero uentry

File /sys/io/uba/uba.c

Routine ubasetup

Problem A page table contains invalid entries.

The routine sets up the UNIBUS adapter map registers from page table entries. The delimiter for the page table is a page table entry containing zero in its page frame bits. While searching the page table, the routine detected a page table entry containing zero in its page frame bits before the delimiter for valid page table entries.

ufs_galloc: dup alloc

File /sys/fs/ufs/ufs_alloc.c

Routine ufs_galloc

Problem A gnode being allocated is not free.

When a gnode is freed, its mode bits are cleared. While attempting to allocate a free gnode from one of the cylinder groups, the routine detected the gnode was not free because its mode bits were set.

Output Indicates the gnode mode bits, the gnode number, and the file system. The format is:

```
mode = <0Xd> inum = <d> fs = <"string">
```

ufs_galloc: ufs_gget returned wrong fs type

File /sys/fs/ufs/ufs_alloc.c

Routine ufs_galloc

Problem A gnode being allocated has the wrong file system type code.

Before this routine allocates a gnode for a file system, it checks the file system type code associated with the gnode. In this case, the routine received the type code and detected it was the wrong type for the gnode.

ufs_gfree

File /sys/fs/ufs/ufs_alloc.c

Routine ufs_gfree

Problem The mode bits of a free gnode indicate the gnode is not free.

When a gnode is freed, its mode bits are cleared. This routine puts freed gnodes on the free list. Before doing so, the routine checks the gnode mode bits. In this case, the routine detected the gnode was not free because its mode bits were still set.

Output Indicates the gnode address and the gnode mode bits. The format is:

```
ufs_gfree: gp <0Xd> mode <0d> should be 0
```

ufs_gfree: freeing free gnode

File /sys/fs/ufs/ufs_alloc.c

Routine ufs_gfree

Problem A gnode being freed is already free.

Before freeing a gnode, the routine checks the used gnode map to determine whether it is free or used. The routine detected the gnode was already marked free.

Output Indicates the device, the gnode number, the file system, and the block. The format is:

```
dev = <0Xd> gno = <d>  fs = <"string"> block <d>
```

ufs_gfree: range

File /sys/fs/ufs/ufs_alloc.c

Routine ufs_gfree

Problem A gnode being freed has an invalid gnode number.

Before freeing a gnode, the routine checks the value of the gnode number passed to it to determine whether the gnode number is valid.

In this case, the routine detected the gnode number was out of range. Its value was greater than or equal to the number of gnodes per cylinder multiplied by the number of cylinder groups.

Output Indicates the device, the gnode number, and the file system. The format is:

```
dev = <0Xd> gno = <d> fs = <"string">
```

ufs_glock: gp type not GT_ULTRIX

File /sys/fs/ufs/ufs_gnode.c

Routine ufs_glock

Problem The file system type code of a gnode is invalid.

Before it unlocks a gnode, this routine checks the file system type code of the specified gnode. In this case, the routine detected the file system type code of the specified gnode was invalid because it was not GT_ULTRIX.

Output Identifies the routine, the gnode address, and the file system type code. The format is:

```
ufs_glock: gp <0Xd> type <d>
```

ufs_grele: gp count bad

File /sys/fs/ufs/ufs_gnode.c

Routine ufs_grele

Problem A gnode being released is already released.

Before releasing a gnode, the routine checks and then clears the reference count of the gnode. While checking the reference count of the gnode, the routine detected it was less than 1. The gnode had already been released.

Output Indicates the routine name, the gnode address, and the gnode number. The format is:

```
ufs_grele: gp <0Xd> (<d>)
```

ufs_gtrunc: newspace

File /sys/fs/ufs/ufs_gnode.c

Routine ufs_gtrunc

Problem No space is returned when a gnode is truncated.

This routine determines the size of the returned space when a gnode is truncated. While doing so, the routine detected that the size of the space returned was zero.

ufs_gtrunc1

File /sys/fs/ufs/ufs_gnode.c

Routine ufs_gtrunc

Problem The indirect block information in an inode does not match that for the gnode.

Before truncating a gnode, this routine matches the indirect block information in the on-disk inode with that in the gnode. The routine detected the indirect block information did not match.

ufs_gtrunc2

File /sys/fs/ufs/ufs_gnode.c

Routine ufs_gtrunc

Problem The direct block information in an inode does not match that for the gnode.

Before truncating a gnode, this routine matches the direct block information in the on-disk inode with that in the gnode. The routine detected the inode direct block information did not match.

ufs_gunlock

File /sys/fs/ufs/ufs_gnode.c

Routine ufs_gunlock

Problem A gnode being unlocked is already unlocked.

Before this routine unlocks a gnode, it checks whether the gnode is already unlocked. In this case, the routine detected the gnode was already unlocked.

Output Identifies the routine, the gnode state, the gnode device, and the gnode number. The format is:

```
ufs_gunlock: gp unlocked, dev <0Xd> gno <d>
```

ufs_gunlock: gp type not GT_ULTRIX

File /sys/fs/ufs/ufs_gnode.c

Routine ufs_gunlock

Problem The file system type code of a gnode is invalid.

Before it unlocks a gnode, this routine checks the file system type code of the specified gnode. The routine detected the file system type code of the gnode was invalid because it was not GT_ULTRIX.

Output Identifies the routine, the gnode address, and the file system type code. The format is:

```
ufs_gunlock: gp <0Xd> type <d>
```

ufs_mount: cannot find root inode

File /sys/fs/ufs/ufs_mount.c

Routine ufs_mount

Problem A file system cannot be mounted because the root gnode cannot be found.

This routine mounts a file system. The routine calls the `ufs_gget` routine to locate the root gnode for the file system. In this case, the routine detects the return from that call is NULL, indicating the `ufs_gget` routine could not locate the root gnode.

ufs_namei: duplicating cache

File /sys/fs/ufs/ufs_namei.c

Routine ufs_namei

Problem A free slot in the namei cache is not free.

While attempting to put a pathname into a namei cache slot, the routine detected the slot was already in use.

ufs_namei: null cache ino

File /sys/fs/ufs/ufs_namei.c

Routine ufs_namei

Problem An inode in the namei cache has a NULL pointer.

While searching the namei cache for a pathname, the routine detected an inode had a NULL pointer.

ufs_rwgp: illegal text reuse

File /sys/fs/ufs/ufs_gnodeops.c

Routine ufs_rwgp

Problem The reference count for a gnode text structure is invalid.

This routine reads and writes gnodes, while keeping track of the reference count for the text structure. The routine detected the reference count for the text structure was invalid because its value was greater than one.

Output Identifies the text address and the gnode address. The format is:

```
textp = <0Xd> gp = <0Xd>
```

ufs_rwgp: messed up gp, xp

File /sys/fs/ufs/ufs_gnodeops.c

Routine ufs_rwgp

Problem A text pointer and gnode pointer pair no longer point to each other.

This routine reads or writes gnodes, while keeping track of the text pointer (to the gnode) and the gnode pointer (to the text). At some point, the routine detected that the pointers no longer point to each other.

Output Identifies the routine, the problem, and the pointers to the gnode and text addresses. The format is:

```
ufs_rwgp: messed up gp, xp
gp <0Xd> xp <0Xd>
```

uipc 1

File /sys/sys/uipc_usrreq.c

Routine uipc_usrreq

Problem The socket type of a user request for socket data is invalid.

The UNIX communications domain supports two types of sockets, stream and datagram. A user request for data received from a socket is supported only for stream sockets, although a user request for data to send to a socket is supported for both stream and datagram socket types. In this case, the routine detected the socket type of a user request for received data was datagram, not stream.

uipc 2

File /sys/sys/uipc_usrreq.c

Routine uipc_usrreq

Problem The socket type of a user request for socket data is invalid.

The UNIX communications domain supports two types of sockets, stream and datagram. In this case, the routine detected that a user request for data received from a socket was invalid because it was neither the stream nor the datagram type.

uipc 3

File /sys/sys/uipc_usrreq.c

Routine uipc_usrreq

Problem A stream socket is not connected to another socket or to a file.

A user request for data to send to a stream socket requires the socket to be connected to another socket or a file before the data can be sent. In this case, the routine checked the `unpcb` structure for the required connection, but discovered none.

uipc 4

File /sys/sys/uipc_usrreq.c

Routine uipc_usrreq

Problem The socket type of a user request for socket data is invalid.

While processing a user request to send data to a stream or datagram socket, the routine detected the socket type was neither stream nor datagram.

uipc 5

File /sys/sys/uipc_usrreq.c

Routine uipc_usrreq

Problem Out-of-band data cannot be sent to a stream socket, because the socket is not connected to a file or another socket.

This routine handles user requests to send out-of-band data to a stream socket. Before the data can be sent, the socket must be connected to a file or to another socket. In this case, the routine detected the socket was not connected to a file or another socket.

unaligned access

File /sys/machine/mips/trap.c

Routine trap()

Problem While running in kernel mode, the system encountered a data access that was not properly aligned. This is a software problem.

unexpected exception

File /sys/machine/mips/locore.s

Routine VEC_unexp

Problem Undefined exception.

The routine received an exception that it does not know how to handle.

Unknown branch instruction

File /sys/machine/mips/trap.c

Routine emulate_branch

Problem The system cannot emulate a branch instruction.

The system called the routine with a nonbranch instruction or an instruction it does not know how to emulate.

unp_connect2

File /sys/sys/uipc_usrreq.c

Routine unp_connect2

Problem A user request to connect a socket has an invalid socket type.

While attempting to connect a socket in the UNIX communications domain, the routine detected the socket type was neither stream nor datagram.

unp_disconnect

File /sys/sys/uipc_usrreq.c

Routine unp_disconnect

Problem A socket being disconnected is not in the `unpcb` data structure.

Before disconnecting a datagram socket, the routine checks the `unpcb` structure for a pointer to this connected socket. In this case, the routine did not find a pointer to the socket in the `unpcb` structure.

unp_externalize

File /sys/sys/uipc_usrreq.c

Routine unp_externalize

Problem There are no file descriptors available for a socket operation.

This routine obtains file descriptors for a datagram type socket with access rights data. While doing so, the routine detected there were no file descriptors available for the operation.

update: Read only file system

File /sys/fs/gfs/gfs_mount.c

Routine update

Problem The file system being updated is a read-only file system.

While updating the mount tables for a file system, the routine detected the file system has been mounted as a read-only file system.

Output Indicates the name of the file system. The format is:

```
fs= <"string"
```

uqdriver: Attempt to open path

File /sys/io/uba/uqserv.c

Routine uq_open_path()

Problem The UQ driver attempted to open a communications path.

This error occurs if the UQ port driver receives a request to open a path. UQ ports do not support initiating connections.

uqdriver: Command ring in invalid state

File /sys/io/uba/uqserv.c

Routine uq_ins_cring()

Problem Command ring in invalid state.

The UQ port driver attempted to place a command in the port command ring and detected that the current command ring entry is in an invalid state.

UQSSP controller failed to reinit

File /sys/io/uba/uda.c

Routine ud_timer

Problem An MSCP disk controller is not reset during hardware initialization.

During a successful hardware initialization sequence, MSCP disk controllers are reset. While checking the software timer for the controller (the UQSSP timer), the routine detected the initialization sequence was unsuccessful because the controller was not reset.

ureadc

File /sys/sys/kern_subr.c

Routine ureadc

Problem There remains data to send, but the I/O count is zero.

The I/O count in the `iovec` structure contains the number of buffers holding data to be sent to or received from a user. The I/O count cannot be zero when data remains to be sent or received. While sending a character to a user, the routine detected that the I/O count was zero.

uwritec

File /sys/sys/kern_subr.c

Routine uwritec

Problem The residual data count and the I/O vector count do not match.

The I/O count in the `iovec` structure contains the number of buffers holding data to be sent to or received from a user. The I/O count cannot be zero when there remains data to be sent or received. While receiving a character from a user, the routine detected that the I/O count was zero or the `uio_resid` value was zero.

VAX state lost...not recoverable

File — /sys/machine/vax/ka8600.c

Routine — ka8600machcheck

Problem — An instruction interrupted by a VAX 8600 machine check cannot be restarted.

After a machine check occurs and the hardware recovers from it, the routine checks the EBOX Control Store register to determine whether the hardware restarted or aborted the interrupted instruction. The routine detected the hardware did not restart the instruction, because the abort bits in the EBOX Control Store register were set. The processor state is lost.

Output — Identifies the machine check type code and other related diagnostic information. See the *Guide to the Error Logger System* for more information.

VAXBI error

File — /sys/io/bi/biinit.c

Routine — bierrors

Problem — Two VAXBI errors occurred within two seconds.

The routine checks the proximity of VAXBI errors while the system is running. While checking the proximity of VAXBI errors, the routine detected two errors within two seconds.

Output — Identifies the name and number of the VAXBI, the node and the error bits for the BI, and the control and status register. The format is:

```
hard error
<"string"> at node <d> error <0Xd> cr <0Xd>
```

vcleanu

File — /sys/vm/vm_mem.c

Routine — vcleanu

Problem — There are no user-list core map (cmap) entries to put on the free list.

This routine puts cmap entires from the user process list on the free list. When the routine is called, there must be cmap entries on the user list. The routine issues this panic when it is called and there are no cmap entries on the user list.

vgetpt

File /sys/vm/vax/pt_machdep.c

Routine vgetpt

Problem A page table for a process is invalid because its size is zero.

This routine gets the page tables for a process. The routine detected the size of the page table was invalid because its size was zero.

vgetsmpt

File /sys/vm/vm_smem.c

Routine vgetsmpt

Problem A shared memory segment is invalid because its size is zero.

This routine gets page tables a process needs for a shared memory segment. In this case, the routine detected the size of a shared memory segment was invalid because its size was zero.

vgetu

File /sys/vm/mips/pt_machdep.c

Routine vgetu

Problem The data swapped into a user area is invalid.

This routine swaps in data to a user area. The routine detected the data was invalid.

vgetu

File /sys/vm/vax/pt_machdep.c

Routine vgetu

Problem The data swapped into a user area is invalid.

This routine swaps in data to a user area. The routine detected the data was invalid.

vgetu bad upage

File /sys/vm/mips/pt_machdep.c

Routine vgetu

Problem A user area pointer does not belong to the process currently in context.

While forking a process, the kernel attempted to copy the user area from the parent to the child process. During this attempt, the routine detected that the passed-in-parent, user area pointer does not belong to the process currently in context.

vinitpt: text pt swap addr 0

File vm/vax/pt_machdep.c

Routine vinitpt

Problem While trying to load the text page table entries from the swap disk, the routine found that the swap disk address of the page tables in text dmap structure was NULL.

vinitpt: text pt swap addr 0

File vm/mips/pt_machdep.c

Routine vinitpt

Problem While trying to load the text page table entries from swap disk, vinitpt found that the swap disk address of the page tables in text dmap structure was NULL.

vinitsmpt

File /sys/vm/vax/pt_machdep.c

Routine vinitsmpt

Problem A shared memory structure is not found in the process structure linked to it.

This routine initializes the shared memory portion of the page table of the process. The routine did not find the segment in the process structure linked to it.

vinitsmpt: p_sm

File /sys/vm/vax/pt_machdep.c

Routine vinitsmpt

Problem NULL pointer to shared memory.

The process argument to this routine has a NULL pointer to shared memory information. The routine initializes process page tables for shared memory using information pointed to by the process structure. However, this information does not exist.

vmemall size

File /sys/vm/vm_mem.c

Routine vmemall

Problem The memory being allocated has a zero size or is greater than the maximum memory allowed.

This routine allocates physical memory. After it receives the size of the memory it is to allocate, the routine does a bounds check on the size. In this case, the routine detected the memory size was either zero or greater than the maximum memory allowed for the process.

vmemfree

File /sys/vm/vm_mem.c

Routine vmemfree

Problem The size of memory being freed is not a multiple of CLSIZE.

This routine frees physical memory. While doing so, the routine detected the size of the memory being freed was not a multiple of CLSIZE

vmemfree vread

File /sys/vm/vm_mem.c

Routine vmemfree

Problem The vread command is not supported by the operating system.

While freeing a memory page, the routine detected an association between a `fpte` and a file descriptor. The association indicated an attempt to execute a `vread` command, which is not supported by the operating system.

vm_system_smget: invalid SMS

File /sys/vm/mips/sm_machdep.c

Routine vm_system_smget

Problem Failure to find the segment data structure.

While the routine was creating or locating a user/system shared memory segment, the routine call to `smget()` succeeded, but its subsequent call to `smconv()` failed to find the segment data structure.

vm_system_smget: invalid SMS

File /sys/vm/vax/sm_machdep.c

Routine vm_system_smget

Problem Failure to find the segment data structure.

While the routine was creating or locating a user/system shared memory segment, the routine call to `smget()` succeeded, but its subsequent call to `smconv()` failed to find the segment data structure.

vpassvm: alloc q->p_sm

File /sys/vm/vm_proc.c

Routine vpassvm

Problem The kernel memory allocator failed to allocate memory for the shared memory information of a process (a km_alloc() problem).

vpassvm: parent has smem, smseg == 0

File /sys/vm/vm_proc.c

Routine vpassvm

Problem A process in a vfork has non-NULL shared memory information, but the system is configured without shared memory.

vrelvm: p_sm

File /sys/vm/vm_proc.c

Routine vrelvm

Problem NULL pointer to shared memory information.

The routine detected a process that indicated it had attached shared memory, but the process has a NULL pointer to shared memory information in the process (proc) structure.

vrelvm rss

File /sys/vm/vm_proc.c

Routine vrelvm

Problem The physical memory resources for a process are released, but the resident set size is not zero.

This routine releases the virtual memory resources associated with a process, such as shared segments and text, data, and stack pages. Then, the routine checks the resident set size for the data and stack of the process to ensure it is zero. In this case, the resident set size for the process was not zero.

Output Identifies the process id, the current resident set size, the text size, and the size of the resident set before the operation began. The format is:

```
p = <0Xd>, p_rssize = <d>, p_textp = <0Xd>, prss_orig = <d>
```

vsalloc: NULL dmap

File vm/vm_pt.c

Routine vsalloc

Problem NULL pointer to dmap structure. Before attempting to allocate swap space for a segment, the routine found a NULL pointer to the dmap information.

vs_bufctl: active pointer null

File /sys/io/uba/uba.c

Routine vs_bufctl

Problem The pointer to the structure for the active driver is NULL.

The VAXstation 2000 and MicroVAX 2000 systems have 16kb of RAM in I/O space that is used for direct memory access between I/O devices. The disk and tape drivers must share exclusive ownership of this hardware RAM buffer.

vs_bufctl: illegal VS_ALLOC returned

File /sys/io/uba/uba.c

Routine vs_bufctl

Problem Illegal VS_ALLOC returned.

The VAXstation 2000 and MicroVAX 2000 systems have 16kb of RAM in I/O space that is used for direct memory access between I/O devices. The disk and tape drivers must share exclusive ownership of this hardware RAM buffer. The routine allows an action parameter to be passed both when it is called and when it is returned. The routine issues this panic when it receives the `VS_ALLOC` parameter as a return value.

vs_bufctl: unknown action

File /sys/io/uba/uba.c

Routine vs_bufctl

Problem The VAXstation 2000 and MicroVAX 2000 systems have 16kb of RAM in I/O space that is used for direct memory access between I/O devices. The disk and tape drivers must share exclusive ownership of this hardware RAM buffer. The routine issues this panic when it has been called with an invalid parameter, or it was returned an invalid parameter.

VS_DEALLOC: no owner

File /sys/io/uba/uba.c

Routine vs_bufctl

Problem No I/O RAM buffer owner.

The VAXstation 2000 and MicroVAX 2000 systems have 16kb of RAM in I/O space that is used for direct memory access between I/O devices. The disk and tape drivers must share exclusive ownership of this hardware RAM buffer. There has been a request from either the disk or tape driver to deallocate ownership of the I/O RAM buffer despite the fact that the buffer is not owned.

VS_DEALLOC: wanted by owner

File /sys/io/uba/uba.c

Routine vs_bufctl

Problem I/O RAM buffer owner is queued to be called back.

The VAXstation 2000 and MicroVAX 2000 systems have 16kb of RAM in I/O space that is used for direct memory access between I/O devices. The disk and tape drivers must share exclusive ownership of this hardware RAM buffer.

If the I/O RAM buffer is in use when requested, the request is queued. When the current owner relinquishes ownership of the RAM buffer, the queued driver is called back. This panic results if the driver that is deallocating the I/O RAM buffer is also queued to be called back.

vsfree: Invalid no. of elems

File vm/vm_pt.c

Routine vsfree

Problem The disk map information is not correct. Before attempting to free the swap space of a segment, the routine found the information about the size of the segment in the dmap structure was not correct.

vsfree: Invalid count

File vm/vm_pt.c

Routine vsfree

Problem This routine releases all the swap space allocated for the segment. After freeing the swap space of the segment, the routine found that either it has released more swap space than allocated or it has more swap space yet to be released.

vsfree: NULL dmap

File vm/vm_pt.c

Routine vsfree

Problem NULL pointer to dmap structure. Before attempting to free the swap space of a segment, the routine found a NULL pointer to the dmap information.

vsxalloc: NULL dmap

File vm/vm_drum.c

Routine vsxalloc

Problem NULL pointer to text dmap structure. Before attempting to allocate swap space for the text segment, the routine found the text structure has a NULL pointer to the dmap information.

vssmalloc: NULL dmap

File vm/vm_drum.c

Routine vssmalloc

Problem NULL pointer to shared memory dmap structure. Before attempting to allocate swap space for the shared memory segment, the routine found the shared memory structure has a NULL pointer to the shared memory dmap information.

vssmfree: NULL dmap

File vm/vm_drum.c

Routine vssmfree

Problem NULL pointer to shared memory dmap structure. Before attempting to free swap space of the shared memory segment, the `vssmalloc` routine found the shared memory structure has a NULL pointer to the shared memory dmap information.

vsswap

File /sys/vm/vm_drum.c

Routine vsswap

Problem The number of memory pages being swapped out is not a multiple of CLSIZE.

This routine swaps out segments of virtual memory. After it receives the number of virtual memory pages it is to swap out, the routine checks the number to ensure it is a multiple of CLSIZE. In this case, the number of virtual memory segments was not a multiple of CLSIZE.

vstodb

File /sys/vm/vm_drum.c

Routine vstodb

Problem The size of a virtual memory block is invalid.

This routine locates contiguous blocks on a disk for a swap out operation. Before the operation, the routine receives several parameters to check. Among these parameters are the base and size for the virtual swap area of the process. The routine detected the base and size parameters were invalid because one of them was less than zero or the size computed for the virtual swap area was greater than the size in the `dmap` structure associated with the process.

vstodb exceeding nswap

File vm/vm_drum.c

Routine vstodb

Problem This routine locates disk blocks for a swap out operation. When it finds a fit for the swap out operation, the routine checks the swap address and the size of the contiguous blocks area to ensure they do not exceed the amount of swap space in the system. In this case, the routine detected the contiguous block area was greater than the amount of swap space in the system.

vtod: shmem

File vm/vm_drum.c

Routine vtod

Problem The size of the shared virtual memory segment is invalid. This routine detected the size computed for the virtual swap area (shared memory segment) was greater than the size in the dmap structure associated with the shared memory segment.

vtod: text

File vm/vm_drum.c

Routine vtod

Problem The size of the text virtual memory segment is invalid. This routine detected the size computed for the virtual swap area (text segment) was greater than the size in the dmap structure associated with the text segment.

VS_WANTBACK: not active

File /sys/io/uba/uba.c

Routine vs_bufctl

Problem No I/O RAM buffer owner.

The VAXstation 2000 and MicroVAX 2000 systems have 16kb of RAM in I/O space that is used for direct memory access between I/O devices. The disk and tape drivers must share exclusive ownership of this hardware RAM buffer. When a driver finishes an I/O request, it needs to allow the other driver access to the I/O RAM buffer.

If the driver that is relinquishing ownership of the I/O RAM buffer still has more I/O requests, it will relinquish the I/O RAM buffer with the intention of being requeued on the buffer. This panic results when a driver tries to relinquish a buffer that it does not own.

vsxfree: NULL dmap

File vm/vm_drum.c

Routine vsxfree

Problem NULL pointer to text dmap structure. Before attempting to free swap space of the text segment, the routine found the text structure has a NULL pointer to the dmap information.

vtod: Can not classify page

File /sys/vm/vm_drum.c

Routine vtod

Problem The user virtual page cannot be classified as text, data, stack, or shared memory.

While attempting to convert a user virtual page number to a disk block number, the routine could not classify the page as text, data, stack, or shared memory.

vtopte: p_sm

File /sys/vm/vm_subr.c

Routine vtopte

Problem NULL pointer to shared memory information.

While attempting to link a process to a shared memory segment, the routine detected the process has a NULL pointer to shared memory information in the process (proc) structure

vtopte SMEM

File /sys/vm/vm_subr.c

Routine vtopte

Problem A shared memory segment cannot be found at a specified memory address.

This routine converts virtual page numbers to page table entry addresses. While searching for a shared memory segment at a specified memory address, the routine could not find the segment.

wakeup

File /sys/sys/kern_synch.c

Routine wakeup

Problem A process being wakened is not sleeping.

This routine wakes up a sleeping process. While doing so, the routine detected the process it was to wake up was not sleeping.

wbaddaddr

File /sys/machine/mips/locore.s

Routine wbadaddr

Problem Bad address length.

The system called the routine with a bad word length. The length is in bytes and can be only 1, 2, or 4.

wdir: blksize

File /sys/fs/ufs/ufs_namei.c

Routine direnter

Problem The DIRBLKSIZ system parameter is greater than the file system fragment size.

This routine checks critical system parameters, such as DIRBLKSIZ, to monitor their size. While monitoring the DIRBLKSIZ system parameter, the routine detected it was greater than the file system fragment size.

wdir: compact1

File /sys/fs/ufs/ufs_namei.c

Routine direnter

Problem There is not enough space for a new directory entry.

Before the routine writes a directory entry, it receives the directory, the gnode to the directory, and space needed for the directory entry. In this case, the routine calculates the current free space and the directory size. Then, the routine detected the entry was the first in a directory block and there was not enough space for it.

wdir: compact2

File /sys/fs/ufs/ufs_namei.c

Routine direnter

Problem There is not enough space for a new directory entry.

Before the routine writes a directory entry, it receives the directory, the gnode to the directory, and space needed for the directory entry. In this case, the routine calculates the size of the current free space and the directory. Then, the routine detected the entry was the second (or subsequent) in a directory block and there was not enough space for it.

wdir: newblk

File /sys/fs/ufs/ufs_namei.c

Routine direnter

Problem The offset for a directory entry is not on a block boundary.

Before the routine writes a directory entry, it receives the directory, the gnode to the directory, and space needed for the directory entry. In addition, the routine uses the count and offset variables stored in the user structure. In this case, the routine detects the count variable is zero, indicating there is no space in the directory. Whenever this happens, the routine then checks the offset variable, which should be on a block boundary. The routine detected the offset variable was not on a block boundary.

wtimo

File /sys/machine/vax/locore.s

Routine wtime

Problem A processor detects an SBIA0 error.

When the processor detects an SBIA0 error caused by a write timeout, it issues an exception and dispatches the error to this routine. In this case, the routine serviced the error by logging it and producing this panic.

Output Identifies the problem at the console subsystem. The format is:

```
write timeout
```

xbi error

File /sys/io/xmi/xbi.c

Routine xbi_check_errs

Problem A nonrecoverable XBI error was detected.

xccdec: text pt swap addr 0

File vm/vm_text.c

Routine xccdec

Problem While trying to save the text page table entries to swap disk, the routine found that the swap disk address of the page tables in text dmap structure was NULL.

xcleanup rssize

File vm/vm_text.c

Routine xcleanup

Problem The physical memory of a shared text segment is released, but its resident set size is not zero.

The routine releases a process use of a shared text segment. After releasing the physical memory of the shared text segment, the routine checks the resident set size to be sure it is zero. The routine detected the resident text size was not zero.

xflush_remote_hash: g_hcount != NULL

File /sys/vm/vm_text.c

Routine xflush_remote_hash

Problem Nonzero final hash count.

The routine has detected an inconsistency in the count of remote text pages that were hashed for a text structure. This routine unhashes these pages, decrementing the count as it handles each page. The final is nonzero.

xflush_remote_hash: x_hcmap == NULL

File /sys/vm/vm_text.c

Routine xflush_remote_hash

Problem NULL pointer to array of remote text hashed pages.

The routine has detected a NULL pointer to an array of remote text hashed pages. Because this routine is called only for remote text, the pointer must be non-NULL.

xfree

File /sys/vm/vm_text.c

Routine xfree

Problem Negative shared text segment.

While attempting to relinquish the use of a shared text segment by a process, the count of processes using the segment is negative.

Output The routine returns the count in the following format:

```
xfree: text 0x%x count bad
```

XMI I/O adapter at wrong address

File /sys/io/xmi/xmiinit.c

Routine xmi_io_space

Problem An MI I/O adapter is at the wrong address.

xrele

File /sys/vm/vm_text.c

Routine xrele

Problem A text pointer in a process does not point to a gnode.

This routine removes a shared text segment from the text table. While doing so, the routine detected the process had a text pointer that did not point to a gnode.

xrepl: lost text

File /sys/vm/vm_text.c

Routine xrepl

Problem Lost process.

During vfork, while replacing one process with either the parent or child process attached to the shared text, the original process cannot be found.

X_RST_HCMAP: hcmap == 0

File /sys/h/gnode.h

Routine G_RST_HCMAP {macro definition}

Problem Hash list corruption.

This routine clears the array element associated with the given page number that is currently being unhashed (used in munhash() and maunhash()). It has detected that the element has already been cleared.

X_RST_HCMAP: page number too large

File /sys/h/gnode.h

Routine G_RST_HCMAP {macro definition}

Problem Hash list corruption.

This routine has detected that the page number contained within the coremap (cmap) entry would place the page beyond the end of the text segment.

X_SET_HCMAP: hcmap != 0

File /sys/h/gnode.h

Routine G_SET_HCMAP {macro definition}

Problem Hash list corruption.

This routine fills the array element associated with the given page number that is currently being hashed (used in mhash()). It has detected that the element has already been filled.

xtiin_pcbunbind not lock owner

File in_pcb.c

Routine xtiin_pcbunbind

Problem The socket referenced by an inpcb control block is not locked.

xunlink no text page tables

File /sys/vm/vm_text.c

Routine xunlink

Problem Segment page tables are not found.

When the last process unlinks from the shared text segment, the page tables for that segment are deallocated. This panic is issued when the routine does not find these page tables.

xunlink x_caddr !NULL

File /sys/vm/vm_text.c

Routine xunlink

Problem Process link list for the segment is not NULL.

When the last process unlinks from the shared text segment, the routine checks to ensure that process link list for this segment is NULL. The routine issues this panic if it finds a process pointer.

X_UNLOCK: text not locked

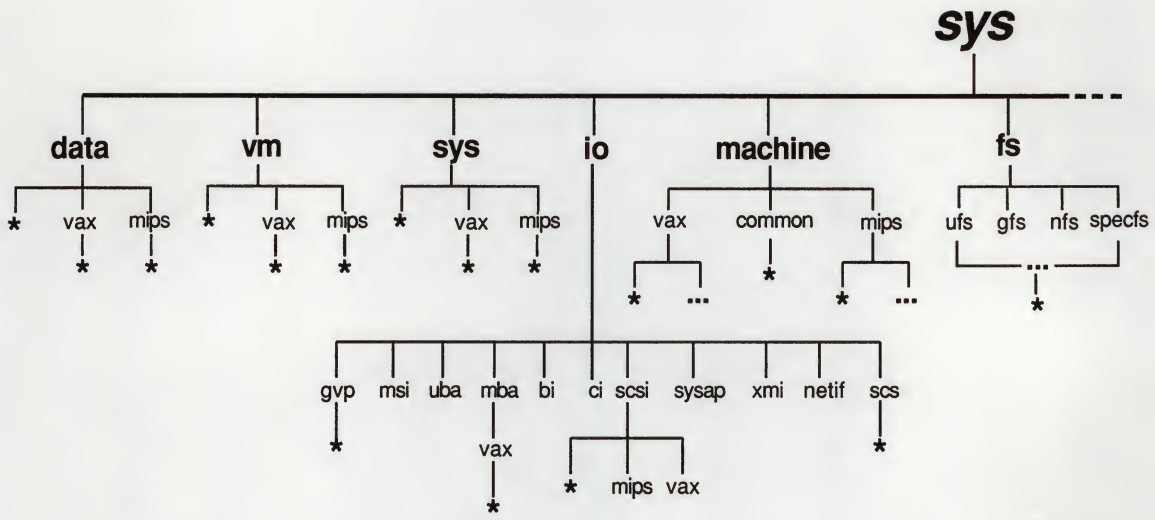
File sys/h/vmmac.h

Routine macro definition X_UNLOCK

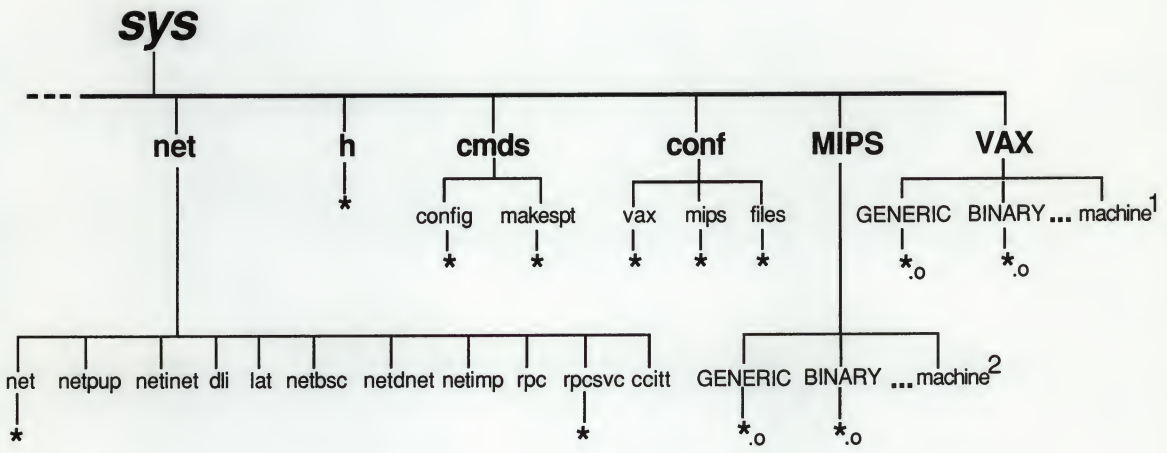
Problem The system has detected that a text segment that it is unlocking is not locked. This violates the locking conventions for text segments.

ULTRIX Kernel Files A

The figure on the next two pages illustrates the directories in the ULTRIX kernel. An asterisk (*) indicates source files at this level.


sys
data
vm
sys
io
machine
fs
vax
mips
vax
mips
vax
mips
vax
common
mips
ufs
gfs
nfs
specfs
gvp
msi
uba
mba
bi
ci
scsi
sysap
xmi
netif
scs
vax
mips
vax

ZK-0096U-R



* indicates source files at this level

1 indicates symbolic link to machine/VAX

2 indicates symbolic link to machine/MIPS

ZK-0097U-R

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-234-1998 using a 1200- or 2400-baud modem from anywhere in the USA, Canada, or Puerto Rico. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital Subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local Digital subsidiary or approved distributor
Internal*	————	SSB Order Processing - WMO/E15 *or* Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

* For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

ULTRIX
Kernel Messages Reference Manual
AA-PBKUA-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? _____

_____

_____

What do you like best about this manual? _____

_____

_____

What do you like least about this manual? _____

_____

_____

Please list errors you have found in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

What version of the software described by this manual are you using? _____

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Email _____ Phone _____

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST–CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3–2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062–9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line

Reader's Comments

ULTRIX
Kernel Messages Reference Manual
AA-PBKUA-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? _____

_____

_____

What do you like best about this manual? _____

_____

_____

What do you like least about this manual? _____

_____

_____

Please list errors you have found in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

What version of the software described by this manual are you using? _____

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Email _____ Phone _____

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST–CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3–2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062–9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line

digital